MYNT EYE D SDK

1.7.2

Generated by Doxygen 1.8.11

Contents

Chapter 1

MYNT EYE D SDK

Chapter 2

Product Introduction

2.1 Product Description

The MYNT Depth utilizes the camera and the motion sensor to provide visually accurate SLAM results with higher precision, lower cost, simpler layout, along with the ability to achieve face and object recognition. The concept of combining binocular and IMU is the leading-edge technology in the current SLAM industry.The depth version of the product has a built-in depth calculation chip that can output depth images without the host computer. At the same time, the product is equipped with leading hardware solutions such as IR active light, IMU six-axis, hardware-level frame synchronization, global shutter, etc., up to 720p/60fps (120°FOV version) of synchronous image information, the recognition distance can reach 15m (50 °FOV version), accuracy up to millimeters (50° FOV version).

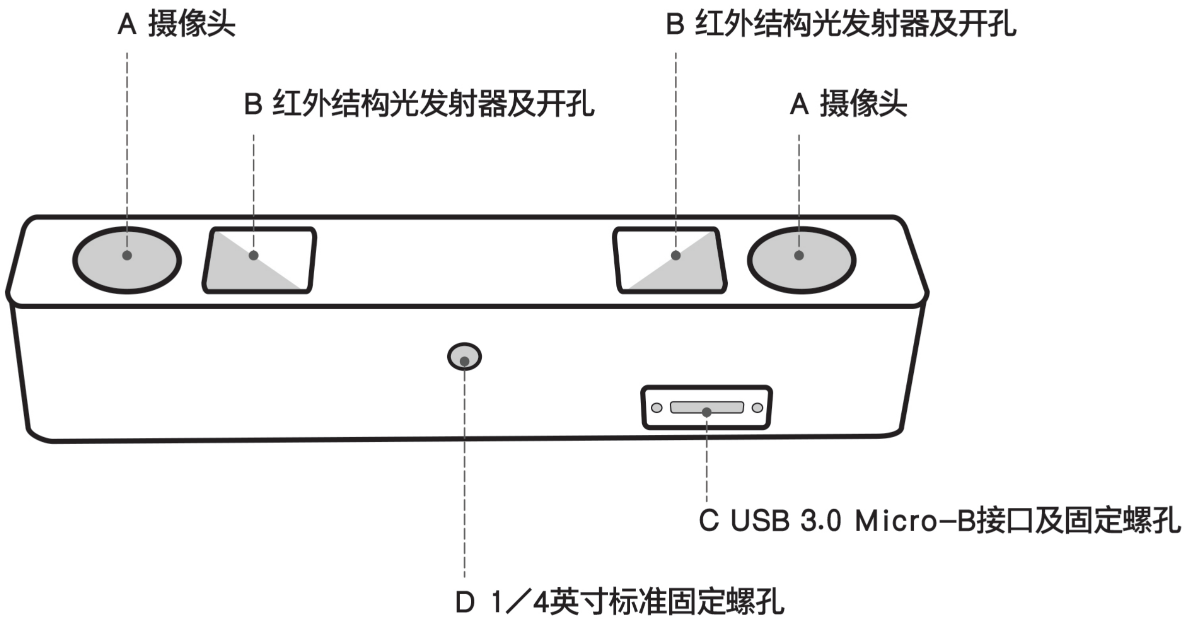
Using camera techniques such as frame synchronization, automatic exposure, and white balance control, the M↩ YNT EYE Depth can produce synchronized image sources with high-precision, which decreases the difficulty of algorithms development, thus increasing efficiency. The Depth comes with six-axis sensor(IMU) and an infrared active light detector (IR). Among them, the six-axis sensor(IMU) can provide complementarity and correction of data from the visual positioning algorithms, and is suitable for visual inertial odometry(VIO) algorithms research to help improve the positioning accuracy. The infrared active light detector (IR) can help solve the problem of identification of objects such as indoor white walls and non-textured objects, as well as enhance the accuracy of image source recognition.The Binocular+IMU scheme provides accurate six-axis complementary data for VSL↩ AM applications and is more accurate and robust than other single solutions. In addition, MYNT EYE Depth also provides a rich SDK interface and VSLAM open source project support, which can help customers quickly integrate solutions, accelerate product development process, and achieve rapid productization and implementation.

As a hardware product for in-depth research and development of stereo vision computing applications, MYNT EYE Depth can be widely used in the field of visual positioning navigation (vSLAM), including visual real-time positioning navigation system of driverless vehicle and robots, visual positioning system of UAV, obstacle avoidance navigation system for driverless Vehicle, Augmented Reality (AR), Virtual Reality (VR), etc. At the same time, it can be used in field of Visual recognition, including Stereoscopic face recognition, three-dimensional object recognition, space motion tracking, three-dimensional gestures and somatosensory recognition. And of course, you can use it for measurement which includes assisted driving system (ADAS), binocular volume calculation, industrial visual screening, etc.

In order to ensure the quality of the output data of the camera products, we have calibrated the binocular and I↩ MU. The product has passed various hardware stability tests, such as high temperature and humidity continuous work and operation, low-temperature dynamic aging, high-temperature operation, low-temperature storage, whole-machine thermal shock, sinusoidal vibration and random vibration tests to ensure the stability and reliability of the product. In addition to the research and development of products and technologies, it can also be directly applied to mass production, accelerating the process from R&D to productization.

2.2 Product Surface

Shell(mm)	PCBA board(mm)
165x31.↩ 5x29.6	149x24



A. Camera:please pay attention to protect the camera sensor lenses, to avoid imaging quality degradation.

B. Infrared structured-light transmitter and outlet: the infrared structured-light can effectively solve the problem associated with the visual positioning calculations of white wall non-textured object(For non-IR version, the outlet is reserved but there is no internal structured-light emitter).

C. USB Micro-B interface and set screw holes: during usage, plug in the USB Micro-B cable and secure it by fastening the set screws to avoid damage to the interface and to ensure stability in connection.

D. $\frac{1}{4}$ inch standardized set screw hole: for fixing the stereo camera to tripods or other devices.

2.3 Support Resolutions

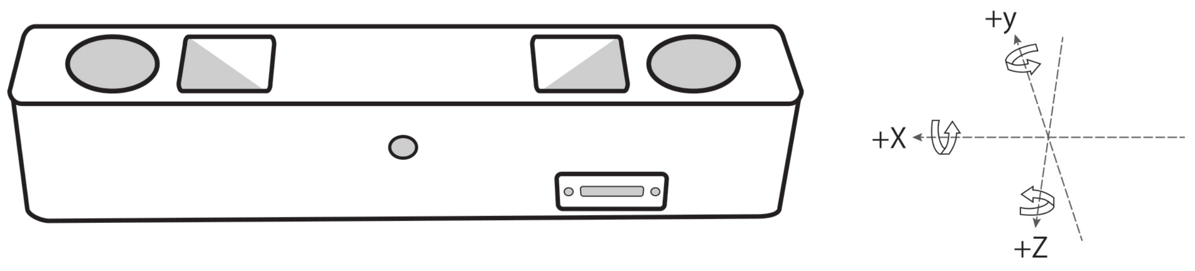
L'=left rectify, L=left, R'=right rectify, R=right, D=depth	interface	color resolution	color fps	depth resolution	depth fps
L'+D	USB3.0	1280x720	60/30/20/10	1280x720	60/30/20/10
L'+D	USB3.0	640x480	60/30	640x480	60/30
L'+R'+D	USB3.0	2560x720	30	1280x720	30
L'+R'+D	USB3.0	1280x480	60/30	640x480	60/30
L+D	USB3.0	1280x720	60/30/20/10	1280x720	60/30/20/10
L+D	USB3.0	640x480	60/30	640x480	60/30
L+R+D	USB3.0	2560x720	30	1280x720	30
L+R+D	USB3.0	1280x480	60/30	640x480	60/30
L+R	USB3.0	2560x720	60/30	not open	null
L'+R'	USB3.0	2560x720	60/30	not open	null
D	USB3.0	not open	null	1280x720	60/30
D	USB3.0	not open	null	640x480	60/30
L+R	USB2.0	2560x720	5	not open	null
L'+R'	USB2.0	2560x720	5	not open	null
L+R	USB2.0	1280x480	15	not open	null
L'+R'	USB2.0	1280x480	15	not open	null
L'+D	USB2.0	1280x720	5	640x720	5
L'+D	USB2.0	640x480	15	320x480	15
L+D	USB2.0	1280x720	5	640x720	5
L+D	USB2.0	640x480	15	320x480	15
L'	USB2.0	1280x720	5	not open	null
L	USB2.0	1280x720	5	not open	null
D	USB2.0	not open	null	640x720	5
D	USB2.0	not open	null	320x480	15
L+R	USB2.0/MJPG	2560x720	5	not open	null
L+R	USB2.0/MJPG	1280x480	15	not open	null
L	USB2.0/MJPG	1280x720	5	not open	null

Note:

- L'=left rectify image, L=left image
- R'=right rectify image, R=right image, D=depth image
- In IR Depth Only mode, frame rate only support 15fps and 30fps.

2.4 IMU Coordinata System

IMU coordinate system is right-handed,the axis directions are as follows:


+y
+X
+Z

Chapter 3

SDK Installation

3.1 Supported Platforms

SDK is built on CMake and can be used cross multiple platforms such as Linux, WIndows,etc.We provide two installation modes:Download pack file and install, Compile and install from source code.

These are the platforms that can be used:

```
* Windows 10
* Ubuntu 18.04/16.04/14.04
* Jetson TX2
* RK3399
```

Tip:

- Ubuntu only support source installation mode.
- Ubuntu 14.04 need to be upgraded to gcc5.

Warning: Due to the requirement of hardware transmission rate, please use the USB 3 interface. In addition, virtual machines have USB driver compatibility problems, thus they are not recommended.

3.2 Quick Start Guide for Linux

1. Install SDK dependencies

1.1 Install OpenCV

If you have installed opencv already or you want use it in ROS, you can skip this part.

1.1.1 Install OpenCV with apt or compile (Choose one)

1.1.1.1 Install OpenCV with apt (Recommend)

```
sudo apt-get install libopencv-dev
```

1.1.1.2 Install OpenCV by Compile

To build and install Opencv, please refer to `Installation in Linux <https://docs.opencv.org/master/d7/d9f/tutorial_linux_install.html>`_ . Alternatively, refer to the command below:

```
[compiler] sudo apt-get install build-essential
[required] sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev
     libswscale-dev
[optional] sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev
     libtiff-dev libjasper-dev libdc1394-22-dev
```

```
git clone https://github.com/opencv/opencv.git
cd opencv/
git checkout tags/3.4.0

cd opencv/
mkdir build
cd build/

cmake ..

make -j4
sudo make install
```

1.2 Install PCL for Point Cloud sample (Optional)

```
sudo apt-get install libpcl-dev libproj-dev libopenni2-dev libopenni-dev
```

1.3 Link libGL.so for TX1/TX2 compile bug (Optional)

```
sudo ln -sf /usr/lib/aarch64-linux-gnu/tegra/libGL.so /usr/lib/aarch64-linux-gnu/libGL.so
```

2. Build SDK

```
git clone https://github.com/slightech/MYNT-EYE-D-SDK.git
cd MYNT-EYE-D-SDK
```

2.1 Init SDK

Note: Because of the problem of device permissions, you must reinsert the camera device after the command is executed and on the same computer, this operation only needs to be done once.

```
make init
```

2.2 Compile SDK

```
make all
```

3. Run Samples

Note:: Open the rectified image by default (Run vio need to raw image, run depth or points cloud need to rectified image.)

1) get_image shows the left and right camera image and colorful depthmap

```
./samples/_output/bin/get_image
```

2) get_depth shows the left camera image, 16UC1 depthmap and depth value(mm) on mouse pointed pixal

```
./samples/_output/bin/get_depth
```

3) get_points shows the left camera image, 16UC1 depthmap and point cloud view

```
./samples/_output/bin/get_points
```

4) get_imu shows motion datas

```
./samples/_output/bin/get_imu
```

5) get_img_params show camera intrinsics and save in file

```
./samples/_output/bin/get_img_params
```

6) get_imu_params show imu intrinsics and save in file

```
./samples/_output/bin/get_imu_params
```

7) get_from_callbacks show image and imu data by callback

```
./samples/_output/bin/get_from_callbacks
```

8) get_all_with_options open device with different options

```
./samples/_output/bin/get_all_with_options
```

4 Install With OpenCV ROS

If you won't use ROS(The Robot Operating System), you can skip this part.

4.1 Install ROS Kinetic

```
cd ~
wget https://raw.githubusercontent.com/oroca/oroca-ros-pkg/master/ros_install.sh && \
chmod 755 ./ros_install.sh && bash ./ros_install.sh catkin_ws kinetic
```

ROS Kinetic will install OpenCV, JPEG.

4.2 Build ROS Wrapper

```
make ros
```

Core:

```
roscore
```

RViz Display:

```
source ./wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d display.launch
```

Publish:

```
source ./wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d mynteye.launch
```

4.3 Build Beta Device ROS Wrapper

```
make ros
```

Core:

```
roscore
```

RViz Display:

```
source ./wrappers/beta_ros/devel/setup.bash
roslaunch mynteye_wrapper_d_beta display.launch
```

Publish:

```
source ./wrappers/beta_ros/devel/setup.bash
roslaunch mynteye_wrapper_d_beta mynteye.launch
```

Subscribe:

```
source ./wrappers/beta_ros/devel/setup.bash
rosrun mynteye_wrapper_d_beta mynteye_listener_d_beta
```

Subscribe:

```
source ./wrappers/beta_ros/devel/setup.bash
rosrun mynteye_wrapper_d_beta mynteye_listener_d_beta
```

5. Package

If you wanna package with specified OpenCV version:

```
cd <sdk>
make cleanall
export OpenCV_DIR=<install prefix>

export OpenCV_DIR=/usr/local
export OpenCV_DIR=$HOME/opencv-2.4.13.3
```

Packaging:

```
cd <sdk>
make pkg
```

6. Clean

```
cd <sdk>
make cleanall
```

3.3 Quick Start Guide for Windows

The following steps are how to install from source codes. If you wanna using prebuilt DLL, please see Windows exe installation .

1. Install Build Tools

1.1 Install Visual Studio

Download Visual Studio 2017 from `https://visualstudio.microsoft.com/` and install

1.2 Install CMake

Download CMake from `https://cmake.org/` and install

1.3 Install MSYS2

1) Download MSYS2 from `http://mirrors.ustc.edu.cn/msys2/distrib/x86_64/` and install

2) Add bin path to System PATH environment variable list

```
C:\msys64\usr\bin
```

3) Install make

```
pacman -Syu
pacman -S make
```

2. Install SDK dependencies

2.1 Install OpenCV

2.1.1 Install OpenCV with Pre-built Libraries (Recommend)

For more details you can reference OpenCV offical document (https://docs.opencv.org/3.4.↩
2/d3/d52/tutorial_windows_install.html)

1) Go to OpenCV Sourceforge page http://sourceforge.net/projects/opencvlibrary/files/opencv-win/
2) Choose a build you want to use and download it. For example 3.4.2/opencv-3.4.2-vc14_vc15.exe 3) Make sure you have admin rights. Unpack the self-extracting archive 4) To finalize the installation, go to set the OpenCV environment variable and add it to the systems path

2.1.2 Set up environment variable

Start up a command window and enter:

∗Change the "D:\OpenCV" to your opencv unpack path∗

```
setx -m OPENCV_DIR D:\OpenCV\Build\x64\vc14\lib     (suggested for Visual Studio 2015 - 64 bit Windows)
setx -m OPENCV_DIR D:\OpenCV\Build\x64\vc15\lib     (suggested for Visual Studio 2017 - 64 bit Windows)
```

Add OpenCV bin path to System PATH environment variable list

```
D:\OpenCV\Build\x64\vc14\bin     (suggested for Visual Studio 2015 - 64 bit Windows)
D:\OpenCV\Build\x64\vc15\bin     (suggested for Visual Studio 2017 - 64 bit Windows)
```

2.2 Install libjpeg-turbo

1) Download libjpeg-turbo from https://sourceforge.net/projects/libjpeg-turbo/files/ and install

2) Add bin path to System PATH environment variable list

```
C:\libjpeg-turbo64\bin
```

2.3 Install PCL for Point Cloud sample (Optional)

Download All-in-one installers (PCL + dependencies) from: https://github.com/PointCloud↩
Library/pcl/releases

3. Build SDK

Open "x64 Native Tools Command Prompt for VS 2017"(适用于 VS 2017 的 x64 本机工具命令提示) command shell

```
git clone https://github.com/slightech/MYNT-EYE-D-SDK.git
cd MYNT-EYE-D-SDK
make all
```

4. Run Samples

> Note: Open the rectified image by default (Run vio need to raw image, run depth or points cloud need to rectified image.)

1) get_image shows the left and right camera image and colorful depthmap

```
.\samples\_output\bin\get_image.bat
```

2) get_depth shows the left camera image, 16UC1 depthmap and depth value(mm) on mouse pointed pixel

```
.\samples\_output\bin\get_depth.bat
```

3) get_points shows the left camera image, 16UC1 depthmap and point cloud view

```
.\samples\_output\bin\get_points.bat
```

4) get_imu shows motion datas

```
.\samples\_output\bin\get_imu
```

5) get_img_params show camera intrinsics and save in file

```
.\samples\_output\bin\get_img_params
```

6) get_imu_params show imu intrinsics and save in file

```
.\samples\_output\bin\get_imu_params
```

7) get_from_callbacks show image and imu data by callback

```
.\samples\_output\bin\get_from_callbacks
```

8) get_all_with_options open device with different options

```
.\samples\_output\bin\get_all_with_options
```

5. Clean

```
cd <sdk>
make cleanall
```

3.4 Windows EXE Installation

Download here: mynteye-d-1.7.1-win-x64-opencv-3.4.3.exe Google Drive, Baidu Pan

After you install the win pack of SDK, there will be a shortcut to the SDK root directory on your desktop.

First, you should plug the MYNT® EYE camera in a USB 3.0 port.

Second, goto the "<SDK_ROOT_DIR>\bin\samples" directory and click "get_image.exe" to run.

Finally, you will see the window that display the realtime frame of the camera.

Generate samples project of Visual Studio 2017

First, you should install Visual Studio 2017 https://visualstudio.microsoft.com/ and CMake https://cmake.org/.

Second, goto the "<SDK_ROOT_DIR>\samples" directory and click "generate.bat" to run.

Finally, you could click `_build\mynteye_samples.sln` to open the samples project.

p.s. The example result of "generate.bat",

```
-- The C compiler identification is MSVC 19.15.26732.1
-- The CXX compiler identification is MSVC 19.15.26732.1
-- Check for working C compiler: C:/Program Files (x86)/Microsoft Visual
     Studio/2017/Community/VC/Tools/MSVC/14.15.26726/bin/Hostx86/x64/cl.exe
-- Check for working C compiler: C:/Program Files (x86)/Microsoft Visual
     Studio/2017/Community/VC/Tools/MSVC/14.15.26726/bin/Hostx86/x64/cl.exe -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual
     Studio/2017/Community/VC/Tools/MSVC/14.15.26726/bin/Hostx86/x64/cl.exe
-- Check for working CXX compiler: C:/Program Files (x86)/Microsoft Visual
     Studio/2017/Community/VC/Tools/MSVC/14.15.26726/bin/Hostx86/x64/cl.exe -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- HOST_ARCH: x86_64
-- Visual Studio >= 2010, MSVC >= 10.0
-- C_FLAGS: /DWIN32 /D_WINDOWS /W3 -Wall -O3
-- CXX_FLAGS: /DWIN32 /D_WINDOWS /W3 /GR /EHsc -Wall -O3
-- Found mynteye: 1.3.6
-- OpenCV ARCH: x64
-- OpenCV RUNTIME: vc15
-- OpenCV STATIC: OFF
-- Found OpenCV: C:/Users/John/AppData/Roaming/Slightech/MYNTEYED/SDK/1.3.6/3rdparty/opencv/build (found
     version "3.4.3")
-- Found OpenCV 3.4.3 in
     C:/Users/John/AppData/Roaming/Slightech/MYNTEYED/SDK/1.3.6/3rdparty/opencv/build/x64/vc15/lib
-- You might need to add
     C:\Users\John\AppData\Roaming\Slightech\MYNTEYED\SDK\1.3.6\3rdparty\opencv\build\x64\vc15\bin to your PATH to be able t
-- Generating executable get_image
-- Generating get_image.bat
-- Generating executable get_depth
-- Generating get_depth.bat
-- Generating executable get_imu
-- Generating get_imu.bat
-- Configuring done
-- Generating done
CMake Warning:
  Manually-specified variables were not used by the project:

    CMAKE_BUILD_TYPE


-- Build files have been written to:
     C:/Users/John/AppData/Roaming/Slightech/MYNTEYED/SDK/1.3.6/samples/_build
Press any key to continue . . .
```

Start using MYNT® EYE Depth SDK with Visual Studio 2017

Goto the "<SDK_ROOT_DIR>\projects\vs2017", see the "README.md".

3.5 ROS Installation

1 Install With OpenCV ROS

If you won't use ROS(The Robot Operating System), you can skip this part.

1.1 Install ROS Kinetic

```
cd ~
wget https://raw.githubusercontent.com/oroca/oroca-ros-pkg/master/ros_install.sh && \
chmod 755 ./ros_install.sh && bash ./ros_install.sh catkin_ws kinetic
```

ROS Kinetic will install OpenCV, JPEG.

1.2 Build ROS Wrapper

```
make ros
```

Core:

```
roscore
```

RViz Display:

```
source ./wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d display.launch
```

Publish:

```
source ./wrappers/ros/devel/setup.bash
roslaunch mynteye_wrapper_d mynteye.launch
```

Subscribe:

```
source ./wrappers/ros/devel/setup.bash
rosrun mynteye_wrapper_d mynteye_listener_d
```

1.3 Build Beta Device ROS Wrapper

```
make ros
```

Core:

```
roscore
```

RViz Display:

```
source ./wrappers/beta_ros/devel/setup.bash
roslaunch mynteye_wrapper_d_beta display.launch
```

Publish:

```
source ./wrappers/beta_ros/devel/setup.bash
roslaunch mynteye_wrapper_d_beta mynteye.launch
```

Subscribe:

```
source ./wrappers/beta_ros/devel/setup.bash
rosrun mynteye_wrapper_d_beta mynteye_listener_d_beta
```

Subscribe:

```
source ./wrappers/beta_ros/devel/setup.bash
rosrun mynteye_wrapper_d_beta mynteye_listener_d_beta
```

3.6 ROS Usage

Compile and run the node according to ROS Installation .

`rostopic list` lists all released nodes:

```
/mynteye/depth/camera_info
/mynteye/depth/image_raw
/mynteye/depth/image_raw/compressed
/mynteye/depth/image_raw/compressed/parameter_descriptions
/mynteye/depth/image_raw/compressed/parameter_updates
/mynteye/depth/image_raw/compressedDepth
/mynteye/depth/image_raw/compressedDepth/parameter_descriptions
/mynteye/depth/image_raw/compressedDepth/parameter_updates
/mynteye/depth/image_raw/theora
/mynteye/depth/image_raw/theora/parameter_descriptions
/mynteye/depth/image_raw/theora/parameter_updates
/mynteye/imu/data_raw
/mynteye/imu/data_raw_processed
/mynteye/left/camera_info
/mynteye/left/image_color
/mynteye/left/image_color/compressed
...
```

`rostopic hz <topic>` checks the data:

```
subscribed to [/mynteye/imu/data_raw]
average rate: 202.806
  min: 0.000s max: 0.021s std dev: 0.00819s window: 174
average rate: 201.167
  min: 0.000s max: 0.021s std dev: 0.00819s window: 374
average rate: 200.599
  min: 0.000s max: 0.021s std dev: 0.00819s window: 574
average rate: 200.461
  min: 0.000s max: 0.021s std dev: 0.00818s window: 774
average rate: 200.310
  min: 0.000s max: 0.021s std dev: 0.00818s window: 974
  ...
```

`rostopic echo <topic>` can print and release data. Please read `rostopic` for more information.

The ROS file is structured like follows:

```
<sdk>/wrappers/ros/
├──src/
  └──mynteye_wrapper_d/
     ├──launch/
     │   ├──display.launch
     │   └──mynteye.launch
     ├──msg/
     ├──rviz/
     ├──src/
      │   ├──mynteye_listener.cc
      │   └──mynteye_wrapper_nodelet.cc
      │   └──mynteye_wrapper_node.cc
      │   └──pointcloud_generatort.cc
      │   └──pointcloud_generator.h
      ├──CMakeLists.txt
      ├──nodelet_plugins.xml
      └──package.xml
```

In `mynteye.launch` ,you can configure `topics` and `frame_ids` ,decide which data to enable, and set the control options. Please set `gravity` to the local gravity acceleration.

```
<arg name="gravity" default="9.8" />
```

Chapter 4

SDK Samples

4.1 Get camera image

Using the `DeviceMode::DEVICE_COLOR` function of the API, you can get color image, or use `Device↩ Mode::DEVICE_ALL` to get color and depth image.

Using `GetStreamData()` to get your data.

Reference code snippet:

```
// Device mode, default DEVICE_ALL
//   DEVICE_COLOR: IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH n
//   DEVICE_DEPTH: IMAGE_LEFT_COLOR n IMAGE_RIGHT_COLOR n IMAGE_DEPTH y
//   DEVICE_ALL:   IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH y
// Note: y: available, n: unavailable, -: depends on #stream_mode
params.dev_mode = DeviceMode::DEVICE_DEPTH;

auto left_color = cam.GetStreamData(ImageType::IMAGE_LEFT_COLOR);
    if (left_color.img) {
     cv::Mat left = left_color.img->To(ImageFormat::COLOR_BGR)->ToMat();
     painter.DrawSize(left, CVPainter::TOP_LEFT);
     painter.DrawStreamData(left, left_color, CVPainter::TOP_RIGHT);
     painter.DrawInformation(left, util::to_string(counter.fps()),
         CVPainter::BOTTOM_RIGHT);
     cv::imshow("left color", left);
```

Complete code samples, see `get_image.cc`.

4.2 Get depth image

Depth images belongs to the upper layer of synthetic data.

You can change `depth_mode` to change the display of the depth image.

```
// Depth mode: colorful(default), gray, raw
params.depth_mode = DepthMode::DEPTH_RAW;
```

Then you can get it through `GetStreamData ()` .In addition, it should be check not be empty before use.

Reference code snippet:

```
auto image_depth = cam.GetStreamData(ImageType::IMAGE_DEPTH);
if (image_depth.img) {
  cv::Mat depth = image_depth.img->To(ImageFormat::DEPTH_RAW)->ToMat();

  cv::setMouseCallback("depth", OnDepthMouseCallback, &depth_region);
  // Note: DrawRect will change some depth values to show the rect.
  depth_region.DrawRect(depth);
  cv::imshow("depth", depth);

  depth_region.ShowElems<ushort>(depth, [](const ushort& elem) {
    return std::to_string(elem);
  }, 80, depth_info);
}
```

The above code uses OpenCV to display the image. When the display window is selected, pressing ESC/Q will end the program.

Complete code examples, see `get_depth.cc`.

4.3 Get point image

Point images belongs to upper layer of synthetic data.You can get it through `GetStreamData()`.It should be check not empty before use.

Sample code snippet:

```
auto image_color = cam.GetStreamData(ImageType::IMAGE_LEFT_COLOR);
auto image_depth = cam.GetStreamData(ImageType::IMAGE_DEPTH);
if (image_color.img && image_depth.img) {
    cv::Mat color = image_color.img->To(ImageFormat::COLOR_BGR)
        ->ToMat();
    painter.DrawSize(color, CVPainter::TOP_LEFT);
    painter.DrawStreamData(color, image_color, CVPainter::TOP_RIGHT);
    painter.DrawInformation(color, util::to_string(counter.fps()),
        CVPainter::BOTTOM_RIGHT);

    cv::Mat depth = image_depth.img->To(ImageFormat::DEPTH_RAW)
        ->ToMat();

    cv::imshow("color", color);

    viewer.Update(color, depth);
}
```

PCL is used to display point images above. Program will close when point image window is closed.

Complete code examples, see `get_points.cc`.

4.4 Get IMU data

You need `EnableMotionDatas()` to enable caching in order to get IMU data from `GetMotionDatas()`.Otherwise, IMU data is only available through the callback interface, see [Get data from callbacks]().

Sample code snippet:

```
auto motion_datas = cam.GetMotionDatas();
if (motion_datas.size() > 0) {
    std::cout << "Imu count: " << motion_datas.size() << std::endl;
    for (auto data : motion_datas) {
        if (data.imu) {
            if (data.imu->flag == MYNTEYE_IMU_ACCEL) {
                counter.IncrAccelCount();
                std::cout << "[accel] stamp: " << data.imu->timestamp
                    << ", x: " << data.imu->accel[0]
                     << ", y: " << data.imu->accel[1]
                     << ", z: " << data.imu->accel[2]
                     << ", temp: " << data.imu->temperature
                     << std::endl;
             } else if (data.imu->flag == MYNTEYE_IMU_GYRO) {
                 counter.IncrGyroCount();
                 std::cout << "[gyro] stamp: " << data.imu->timestamp
                 << ", x: " << data.imu->gyro[0]
                 << ", y: " << data.imu->gyro[1]
                 << ", z: " << data.imu->gyro[2]
                 << ", temp: " << data.imu->temperature
                 << std::endl;
             } else {
                 std::cerr << "Imu type is unknown" << std::endl;
             }
         } else {
           std::cerr << "Motion data is empty" << std::endl;
         }
     }
     std::cout << std::endl;
}
```

OpenCV is used to display image and data. When window is selected, press ESC/Q to exit program.

Complete code examples, see `get_imu.cc`.

4.5 Get data from callbacks

API offers function `SetStreamCallback()` and `SetMotionCallback()` to set callbacks for various data.

Reference code snippet:

```
cam.SetImgInfoCallback([](const std::shared_ptr<ImgInfo>& info) {
    std::cout << "  [img_info] fid: " << info->frame_id
        << ", stamp: " << info->timestamp
        << ", expos: " << info->exposure_time << std::endl
        << std::flush;
});
for (auto&& type : types) {
    // Set stream data callback
    cam.SetStreamCallback(type, [](const StreamData& data) {
     std::cout << "  [" << data.img->type() << "] fid: "
         << data.img->frame_id() << std::endl
         << std::flush;
     });
}

// Set motion data callback
cam.SetMotionCallback([](const MotionData& data) {
     if (data.imu->flag == MYNTEYE_IMU_ACCEL) {
         std::cout << "[accel] stamp: " << data.imu->timestamp
         << ", x: " << data.imu->accel[0]
         << ", y: " << data.imu->accel[1]
         << ", z: " << data.imu->accel[2]
         << ", temp: " << data.imu->temperature
         << std::endl;
     } else if (data.imu->flag == MYNTEYE_IMU_GYRO) {
         std::cout << "[gyro] stamp: " << data.imu->timestamp
         << ", x: " << data.imu->gyro[0]
         << ", y: " << data.imu->gyro[1]
         << ", z: " << data.imu->gyro[2]
         << ", temp: " << data.imu->temperature
         << std::endl;
     }
     std::cout << std::flush;
});
```

OpenCV is used to display images and data above. When the window is selected, pressing ESC/Q will exit program.

Complete code examples, see `get_from_callbacks.cc`.

4.6 Get different types of image by options

`get_all_with_options` sample can add different options to control device.

`get_all_with_options -h`:

```
Open device with different options.

Options:
  -h, --help            show this help message and exit
  -m, --imu             Enable imu datas

  Open Params:
    The open params

     -i INDEX, --index=INDEX
                         Device index
     -f RATE, --rate=RATE
                         Framerate, range [0,60], [30](STREAM_2560x720),
                         default: 10
     --dev-mode=MODE       Device mode, default 2 (DEVICE_ALL)
                           0: DEVICE_COLOR, left y right - depth n
                           1: DEVICE_DEPTH, left n right n depth y
                           2: DEVICE_ALL,   left y right - depth y
                           Note: y: available, n: unavailable, -: depends on
                         stream mode
     --cm=MODE           Color mode, default 0 (COLOR_RAW)
                           0: COLOR_RAW, color raw
                           1: COLOR_RECTIFIED, color rectified
     --dm=MODE           Depth mode, default 2 (DEPTH_COLORFUL)
                           0: DEPTH_RAW
                           1: DEPTH_GRAY
                           2: DEPTH_COLORFUL
     --sm=MODE           Stream mode of color & depth,
                         default 2 (STREAM_1280x720)
                           0: STREAM_640x480, 480p, vga, left
                           1: STREAM_1280x480, 480p, vga, left+right
                           2: STREAM_1280x720, 720p, hd, left
                           3: STREAM_2560x720, 720p, hd, left+right
     --csf=MODE          Stream format of color,
                         default 1 (STREAM_YUYV)
                           0: STREAM_MJPG
                           1: STREAM_YUYV
     --dsf=MODE          Stream format of depth,
                         default 1 (STREAM_YUYV)
                           1: STREAM_YUYV
     --ae                Enable auto-exposure
     --awb               Enable auto-white balance
     --ir=VALUE          IR intensity, range [0,6], default 0
     --ir-depth          Enable ir-depth-only

   Feature Toggles:
     The feature toggles

     --proc=MODE         Enable process mode, e.g. imu assembly, temp_drift
                           0: PROC_NONE
                           1: PROC_IMU_ASSEMBLY
                           2: PROC_IMU_TEMP_DRIFT
                           3: PROC_IMU_ALL
     --img-info          Enable image info, and sync with image
```

e.g. `./samples/_output/bin/get_all_with_options -f 60 --dev-mode=0 --sm=2` displays 1280x720 60fps left unrectified image.

Complete code samples，see `get_all_with_options` .

4.7 Get image calibration parameters

Use `GetStreamIntrinsics()` and `GetStreamExtrinsics()` to get image calibration parameters.

Reference code snippet：

```
auto vga_intrinsics = cam.GetStreamIntrinsics(StreamMode::STREAM_1280x480, &in_ok);
auto vga_extrinsics = cam.GetStreamExtrinsics(StreamMode::STREAM_1280x480, &ex_ok);
std::cout << "VGA Intrinsics left: {" << vga_intrinsics.left << "}" << std::endl;
std::cout << "VGA Intrinsics right: {" << vga_intrinsics.right << "}" << std::endl;
std::cout << "VGA Extrinsics left to right: {" << vga_extrinsics << "}" << std::endl;
out << "VGA Intrinsics left: {" << vga_intrinsics.left << "}" << std::endl;
out << "VGA Intrinsics right: {" << vga_intrinsics.right << "}" << std::endl;
out << "VGA Extrinsics left to right: {" << vga_extrinsics << "}" << std::endl;
```

The result will be saved in the current file directory.Reference result on Linux:

```
VGA Intrinsics left: {width: [640], height: [480], fx: [358.45721435546875000], fy:
      [359.53115844726562500], cx: [311.12109375000000000], cy: [242.63494873046875000]coeffs: [-0.28297042846679688,
      0.06178283691406250, -0.00030517578125000, 0.00218200683593750, 0.00000000000000000]}
VGA Intrinsics right: {width: [640], height: [480], fx: [360.13885498046875000], fy:
      [360.89624023437500000], cx: [325.11029052734375000], cy: [251.46371459960937500]coeffs: [-0.30667877197265625,
      0.08611679077148438, -0.00030136108398438, 0.00155639648437500, 0.00000000000000000]}
VGA Extrinsics left to right: {rotation: [0.99996054172515869, 0.00149095058441162, 0.00875246524810791,
      -0.00148832798004150, 0.99999880790710449, -0.00030362606048584, -0.00875294208526611, 0.00029063224792480,
      0.99996161460876465], translation: [-120.36341094970703125, 0.00000000000000000, 0.00000000000000000]}
```

Complete code examples, see `get_img_params.cc`.

4.8 Get IMU calibration parameters

Use `GetMotionIntrinsics()` and `GetMotionExtrinsics` to get current IMU calibration parameters.

Reference code snippet:

```
auto intrinsics = cam.GetMotionIntrinsics(&in_ok);
std::cout << "Motion Intrinsics: {" << intrinsics << "}" << std::endl;
out << "Motion Intrinsics: {" << intrinsics << "}" << std::endl;
```

The result will be saved in the current file directory.Reference result on Linux:

```
Motion Intrinsics: {accel: {scale: [1.00205999990004191, 0.00000000000000000, 0.00000000000000000,
      0.00000000000000000, 1.00622999999999996, 0.00000000000000000, 0.00000000000000000, 0.00000000000000000,
      1.00171999999999994], assembly: [1.00000000000000000, 0.00672262000000000, -0.00364474000000000, 0.00000000000000000,
      1.00000000000000000, 0.00101348000000000, -0.00000000000000000, 0.00000000000000000, 1.00000000000000000,
      1.00000000000000000], drift: [0.00000000000000000, 0.00000000000000000, 0.00000000000000000], noise:
      [0.00000000000000000, 0.00000000000000000, 0.00000000000000000], bias: [0.00000000000000000, 0.00000000000000000,
      0.00000000000000000], x: [0.00856165620000000, -0.00009840052800000], y: [0.05968393300000000,
      -0.00130967680000000], z: [0.01861442050000000, -0.00016033523000000]}, gyro: {scale: [1.00008999999999992,
      0.00000000000000000, 0.00000000000000000, 0.00000000000000000, 0.99617599999999995, 0.00000000000000000, 0.000000000000
      0.00000000000000000, 1.00407000000000002], assembly: [1.00000000000000000, -0.00700362000000000,
      -0.00326206000000000, 0.00549571000000000, 1.00000000000000000, 0.00224867000000000, 0.00236088000000000,
      0.00044507800000000, 1.00000000000000000, 1.00000000000000000], drift: [0.00000000000000000, 0.00000000000000000,
      0.00000000000000000], noise: [0.00000000000000000, 0.00000000000000000, 0.00000000000000000], bias:
      [0.00000000000000000, 0.00000000000000000, 0.00000000000000000], x: [0.18721455299999998, 0.00077411070000000], y:
      [0.60837032000000002, -0.00939702710000000], z: [-0.78549276000000001, 0.02584820200000000]}}
```

Complete code examples, see `get_imu_params.cc`.

4.9 Set open parameters

Set the resolution of image

Using the `params.stream_mode` parameter, you can set the resolution of the image.

> Attention: Now image resolution supports 4 types: 640X480，1280x720 for single camera. 1280x480， 2560x720 for left and right camera.

Reference code snippet:

```
// Stream mode: left color only
// params.stream_mode = StreamMode::STREAM_640x480;  // vga
// params.stream_mode = StreamMode::STREAM_1280x720;  // hd
// Stream mode: left+right color
// params.stream_mode = StreamMode::STREAM_1280x480;  // vga
params.stream_mode = StreamMode::STREAM_2560x720;  // hd
```

Set the frame rate of image

Using the `params.framerate` parameter, you can set the frame rate of image.

> Attention
>
> - The effective fps of the image(0-60)
> - The effective fps of the image in 2560x720 resolution (30)

Reference code snippet:

```
// Framerate: 30(default), [0,60], [30](STREAM_2560x720)
params.framerate = 30;
```

Set color mode

Using the `params.color_mode` parameter， you can set the color mode of image.

`COLOR_RAW` is original image， `COLOR_RECTIFIED` is rectified image.

Reference code snippet:

```
// Color mode: raw(default), rectified
// params.color_mode = ColorMode::COLOR_RECTIFIED;
```

Set depth mode

Using the `params.depth_mode` parameter， you can set the depth mode.

`DEPTH_COLORFUL` is colorful depth image， `DEPTH_GRAY` is grey depth image， `DEPTH_RAW` is original depth image。

Reference code snippet:

```
// Depth mode: colorful(default), gray, raw
// params.depth_mode = DepthMode::DEPTH_GRAY;
```

Enable auto exposure and auto white balance

Set `params.state_ae` and `params.state_awb` to `true` , you can enable auto exposure and auto white balance.

By default auto exposure and auto white balance are enabled，if you want to disable，you can set parameters to `false` .

Reference code snippet:

```
// Auto-exposure: true(default), false
// params.state_ae = false;

// Auto-white balance: true(default), false
// params.state_awb = false;
```

Enable IR and its adjustments function

Using the `params.ir_intensity` parameter，you can set IR's intensity of image. Enabling IR is setting `params.ir_intensity` greater than 0. The greater the value, the greater the IR's intensity.(max is 10).

Reference code snippet:

```
// Infrared intensity: 0(default), [0,10]
params.ir_intensity = 4;
```

Enable IR Depth Only

Using the `params.ir_depth_only` parameter，you can set IR Depth Only function. This is disabled by default.

Attention

- This function doesn't work on 15 frame rate below.

Reference code snippet:

```
// IR Depth Only: true, false(default)
// Note: IR Depth Only mode support frame rate between 15fps and 30fps.
//     When dev_mode != DeviceMode::DEVICE_ALL,
//       IR Depth Only mode not be supported.
//     When stream_mode == StreamMode::STREAM_2560x720,
//       frame rate only be 15fps in this mode.
//     When frame rate less than 15fps or greater than 30fps,
//       IR Depth Only mode will be not available.
// params.ir_depth_only = false;
```

Adjust colour depth value

Using the `params.colour_depth_value` parameter, The value is 1000 by default.

Reference code snippet:

```
// Colour depth image, default 1000. [0, 16384]
// params.colour_depth_value = 1000;
```

Reference running results on Linux:

```
Open device: 0, /dev/video1

D/eSPDI_API: SetPropertyValue control=7 value=0D/eSPDI_API: SetPropertyValue control=7 value=35D/eSPDI_API:
      SetPropertyValue control=7 value=1-- Auto-exposure state: enabled
D/eSPDI_API: SetPropertyValue control=7 value=0D/eSPDI_API: SetPropertyValue control=7 value=12D/eSPDI_API:
      SetPropertyValue control=7 value=1-- Auto-white balance state: enabled
-- Framerate: 5
D/eSPDI_API: SetPropertyValue control=7 value=4 SetDepthDataType: 4
-- Color Stream: 1280x720 YUYV
-- Depth Stream: 1280x720 YUYV

D/eSPDI_API: SetPropertyValue control=7 value=0D/eSPDI_API: SetPropertyValue control=7 value=3D/eSPDI_API:
      SetPropertyValue control=7 value=4
-- IR intensity: 4
D/eSPDI_API: CVideoDevice::OpenDevice 1280x720 fps=5

Open device success
```

Complete code samples， see `get_image` .

4.10 Camera control parameters API

Open or close auto exposure

```
/** Auto-exposure enabled or not  default enabled*/
bool AutoExposureControl(bool enable);    see "camera.h"
```

Open or close auto white balance

```
/** Auto-white-balance enabled or not  default enabled*/
bool AutoWhiteBalanceControl(bool enable);    see "camera.h"
```

Set infrared(IR) intensity

```
/** set infrared(IR) intensity [0, 10] default 4*/
void SetIRIntensity(const std::uint16_t &value);     see "camera.h"
```

Set global gain

Note:: You have to close auto exposure first.

```
/** Set global gain [1 - 16]
 * value -- global gain value
 * */
void SetGlobalGain(const float &value);    see "camera.h"
```

Set the exposure time

Note:: You have to close auto exposure first.

```
/** Set exposure time [1ms - 2000ms]
 * value -- exposure time value
 * */
void SetExposureTime(const float &value);    see "camera.h"
```

Chapter 5

SDK Tools

5.1 Analyze IMU data

The SDK provides the script imu_analytics.py for IMU analysis. The tool details can be seen in tools/README.md .

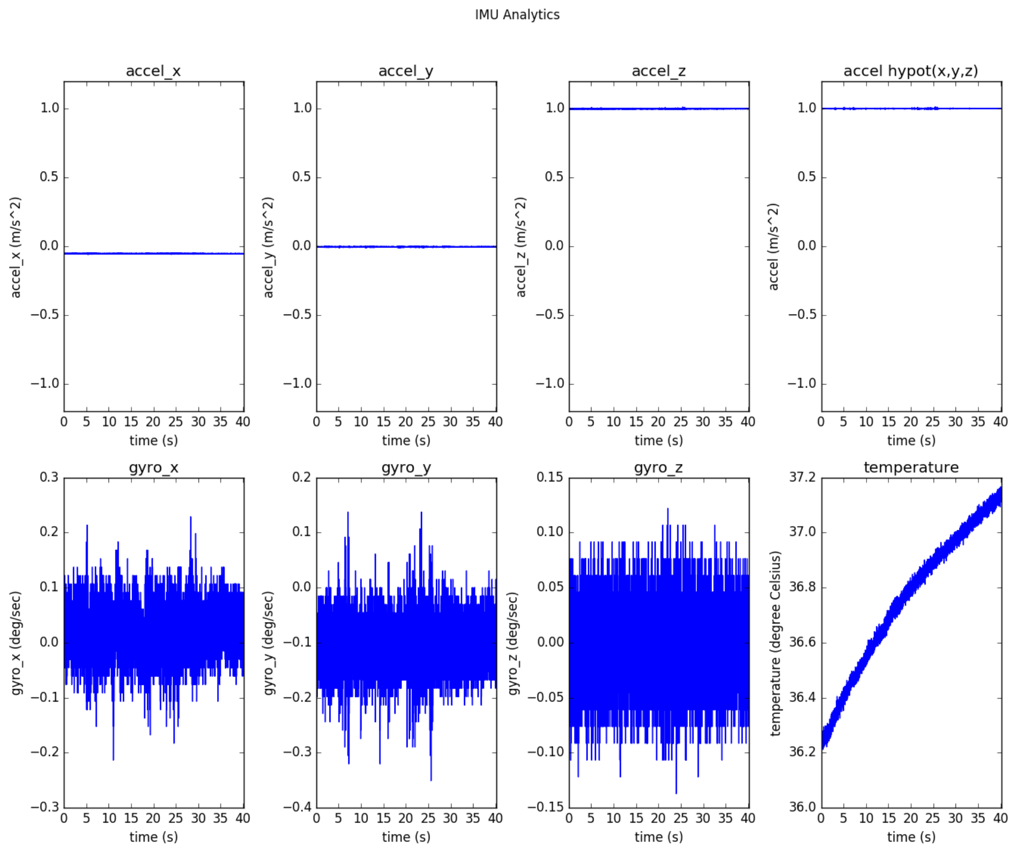
Reference to run commands on Linux:

```
$ python tools/analytics/imu_analytics.py -i dataset -c tools/config/mynteye/mynteye_config.yaml
       -al=-1.2,1.2 -gl= -gdu=d -gsu=d -kl=
```

Reference to results on Linux:

```
$ python tools/analytics/imu_analytics.py -i dataset -c tools/config/mynteye/mynteye_config.yaml
       -al=-1.2,1.2 -gl= -gdu=d -gsu=d -kl=
imu analytics ...
  input: dataset
  outdir: dataset
  gyro_limits: None
  accel_limits: [(-1.2, 1.2), (-1.2, 1.2), (-1.2, 1.2), (-1.2, 1.2)]
  time_unit: None
  time_limits: None
  auto: False
  gyro_show_unit: d
  gyro_data_unit: d
  temp_limits: None
open dataset ...
  imu: 20040, temp: 20040
  timebeg: 4.384450, timeend: 44.615550, duration: 40.231100
save figure to:
  dataset/imu_analytics.png
imu analytics done
```

The analysis result graph will be saved in the data set directory. as follows:



In addition, the script specific options can be executed -h:

```
$ python tools/analytics/imu_analytics.py -h
```

5.2 Analyze time stamps

SDK provides a script for timestamp analysis stamp_analytics.py . Tool details are visible in tools/README.md .

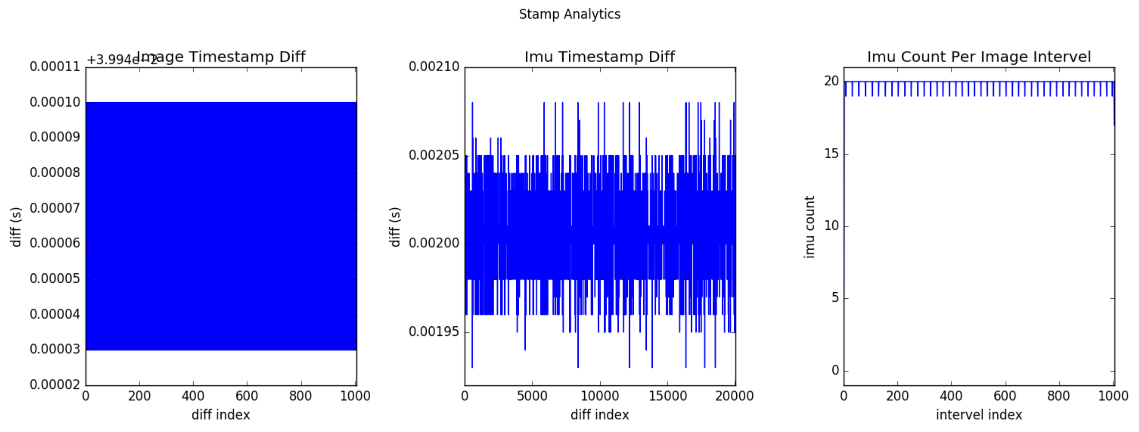
Reference run commands on Linux:

```
$ python tools/analytics/stamp_analytics.py -i dataset -c tools/config/mynteye/mynteye_config.yaml
```

Reference to results on Linux:

```
$ python tools/analytics/stamp_analytics.py -i dataset -c tools/config/mynteye/mynteye_config.yaml
stamp analytics ...
  input: dataset
  outdir: dataset
open dataset ...
save to binary files ...
  binimg: dataset/stamp_analytics_img.bin
  binimu: dataset/stamp_analytics_imu.bin
  img: 1007, imu: 20040

rate (Hz)
  img: 25, imu: 500
sample period (s)
  img: 0.04, imu: 0.002

diff count
  imgs: 1007, imus: 20040
  imgs_t_diff: 1006, imus_t_diff: 20039

diff where (factor=0.1)
  imgs where diff > 0.04*1.1 (0)
  imgs where diff < 0.04*0.9 (0)
  imus where diff > 0.002*1.1 (0)
  imus where diff < 0.002*0.9 (0)

image timestamp duplicates: 0

save figure to:
  dataset/stamp_analytics.png
stamp analytics done
```

The analysis result graph will be saved in the dataset directory. as follow:



In addition, the script specific options can be executed -h to understand:

```
$ python tools/analytics/stamp_analytics.py -h
```

5.3 Record data sets

The SDK provides the tool record for recording data sets. Tool details can be seen in tools/README.md .

Reference run command on Linux:

```
./tools/_output/bin/dataset/record
```

Reference run command on Windows:

```
.\tools\_output\bin\dataset\record.bat
```

Reference run results on Linux:

```
$ ./tools/_output/bin/dataset/record
Saved 1007 imgs, 20040 imus to ./dataset
I0513 21:29:38.608772 11487 record.cc:118] Time beg: 2018-05-13 21:28:58.255395, end: 2018-05-13
      21:29:38.578696, cost: 40323.3ms
I0513 21:29:38.608853 11487 record.cc:121] Img count: 1007, fps: 24.9732
I0513 21:29:38.608873 11487 record.cc:123] Imu count: 20040, hz: 496.983
```

Results save into `<workdir>/dataset` by default. You can also add parameter, select other directory to save.

Record contents:

```
<workdir>/
└─dataset/
   ├─left/
   │  ├─stream.txt  # Image infomation
   │  └─...
   ├─right/
   │  ├─stream.txt  # Image information
   │  └─...
   └─motion.txt  # IMU information
```

Copyright 2018. MYNTEYE

5.4 Save device infomation and parameters

The SDK provides a tool `save_all_infos` for save information and parameters.

Reference commands:

```
./tools/_output/bin/writer/save_all_infos

# Windows
.\tools\_output\bin\writer\save_all_infos.bat
```

Reference result on Linux:

```
I/eSPDI_API: eSPDI: EtronDI_Init
Device descriptors:
  name: MYNT-EYE-D1000
  serial_number: 203837533548500F002F0028
  firmware_version: 1.0
  hardware_version: 2.0
  spec_version: 1.0
  lens_type: 0000
  imu_type: 0000
  nominal_baseline: 120
```

Result save into `<workdir>/config` by default. You can also add parameters to select other directory for save.

Saved contents:

```
<workdir>/
└─config/
   └─SN0610243700090720/
      ├─device.info
      └─imu.params
```

Complete code samples, see `save_all_infos` .

5.5 Write IMU parameters

SDK provides the tool `imu_params_writer` to write IMU parameters.

Information about how to get IMU parameters, please read [Get IMU calibration parameters]() .

Reference commands:

```
./tools/_output/bin/writer/imu_params_writer tools/writer/config/imu.params

# Windows
.\tools\_output\bin\writer\imu_params_writer.bat tools\writer\config\imu.params
```

The path of parameters file can be found in `tools/writer/config/imu.params` . If you calibrated the parameters yourself, you can edit the file and run above commands to write them into the device.

Warning

- Please don't override parameters, you can use `save_all_infos` to backup parameters.

Complete code samples， see `imu_params_writer` .

5.6 Update HID Firmware

Get Firmware

Latest firmware: mynteye-d-hid-firmware-1.0.bin `Google Drive`, `Baidu Pan`

Compile SDK Tools

```
cd <sdk>
make tools
```

Update Firmware

```
./tools/_output/bin/writer/device_hid_update <firmware-file-path>
```

5.7 Update Camera Firmware

Note:: This tool does not support beta device upgrade.

Get Firmware

Latest firmware: SICI-B12-B0135P-016-005-ISO_Plugout2M(Interleave).bin `Google Drive`, `Baidu Pan`

Get Update Tool

Latest tool: EtronWriter_1.0.6.zip `Google Drive`, `Baidu Pan`

Update Firmware

Note:: Please follow the steps to upgrade firmware.(Otherwise, the camera calibration parameters will be lost.)

1, Select camera device.

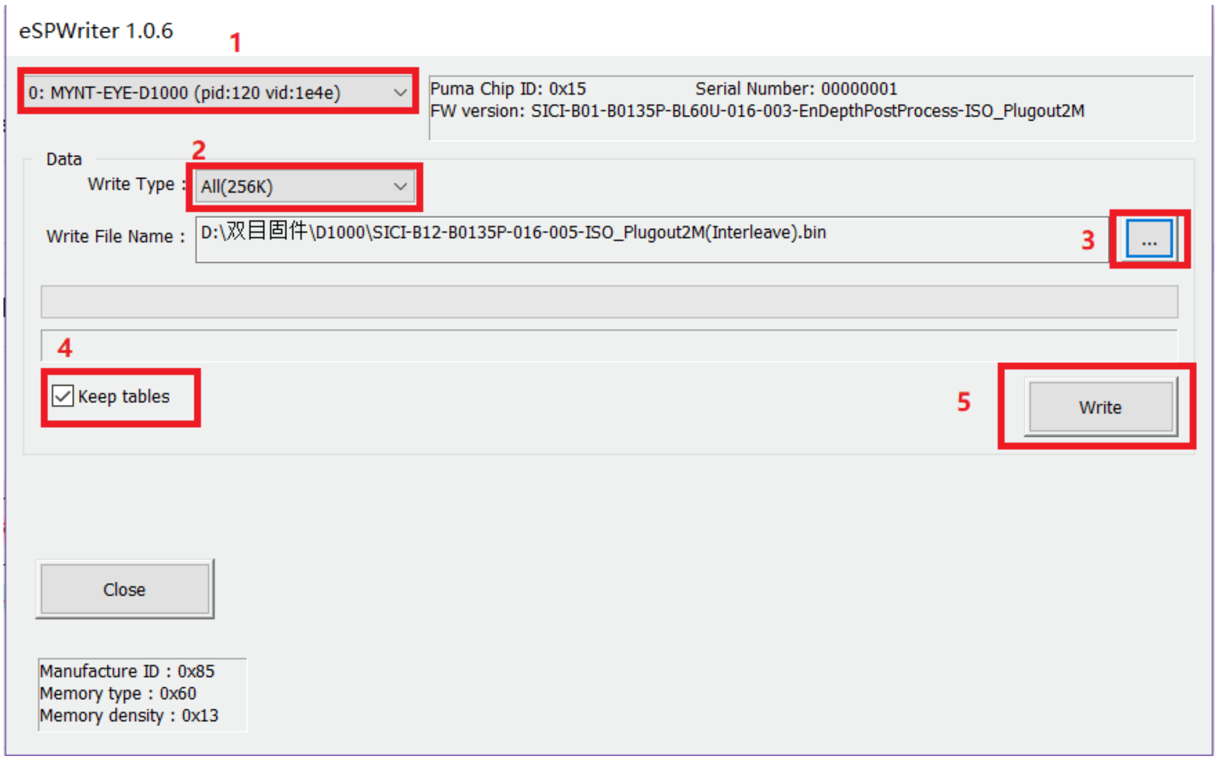
2, Select data type(256KB).

3, Select camera firmware.

4, Select `Keep tables` (in order to keep calibration parameters).

5, Click `Write`.

Use the tool according to diagram:



eSPWriter 1.0.6

0: MYNT-EYE-D1000 (pid:120 vid:1e4e)

Puma Chip ID: 0x15 Serial Number: 00000001
FW version: SICI-B01-B0135P-BL60U-016-003-EnDepthPostProcess-ISO_Plugout2M

Data
Write Type : All(256K)
Write File Name : D:\双目固件\D1000\SICI-B12-B0135P-016-005-ISO_Plugout2M(Interleave).bin

☑ Keep tables

Manufacture ID : 0x85
Memory type : 0x60
Memory density : 0x13

Chapter 6

Project Demos

6.1 How to use SDK with Visual Studio 2017

This tutorial will create a project with Visual Studio 2017 to start using SDK.

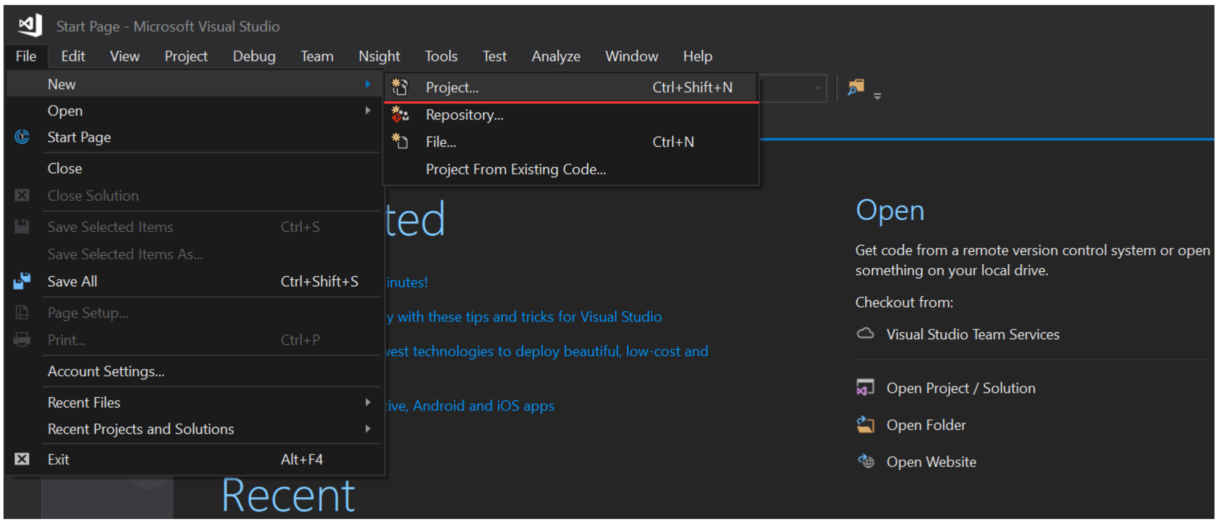
> You could find the project demo in `<sdk>/platforms/projects/vs2017` directory.

Preparation

- Windows: install the win pack of SDK

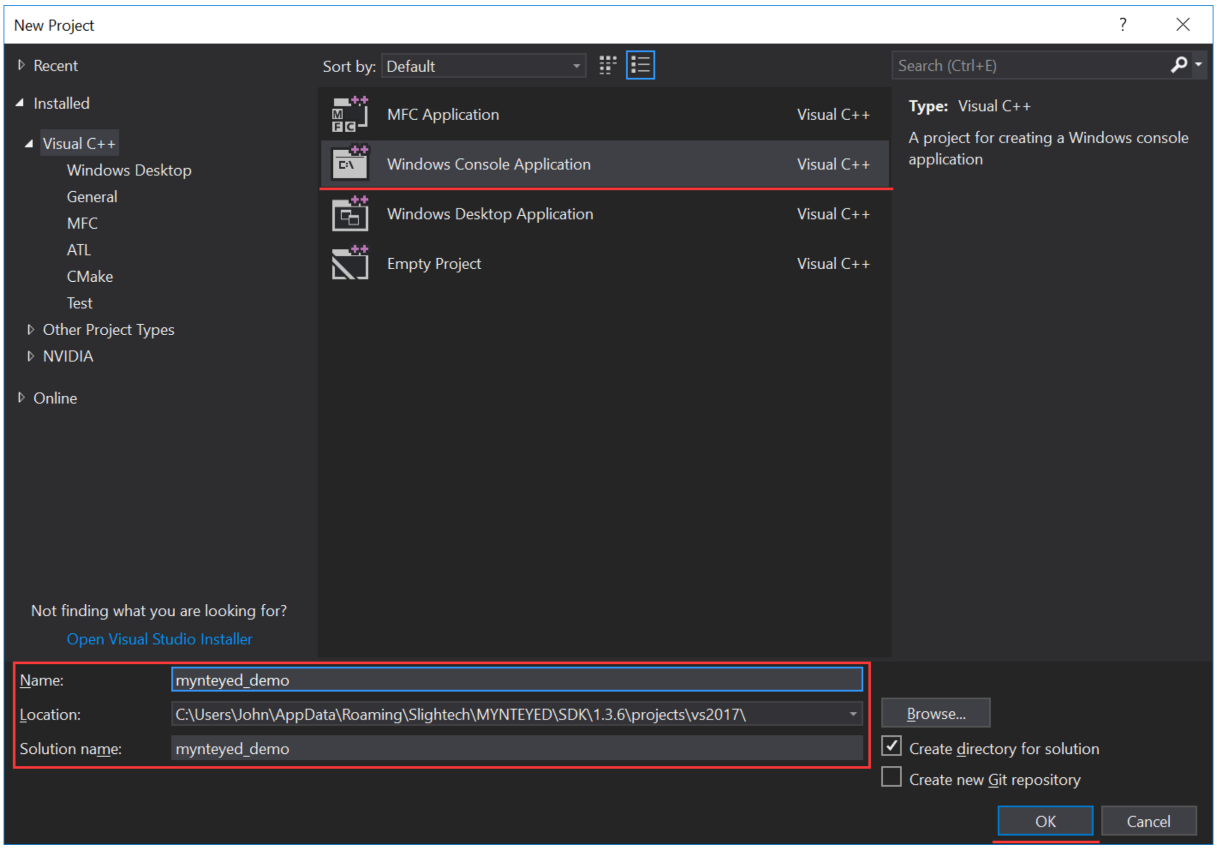
Create Project

Open Visual Studio 2017, then `File > New > Project`,



Get code from a remote version control system or open something on your local drive.

Select "Windows Console Application", set the project's name and location, click "OK",



Type: Visual C++

A project for creating a Windows console application

Name: mynteyed_demo

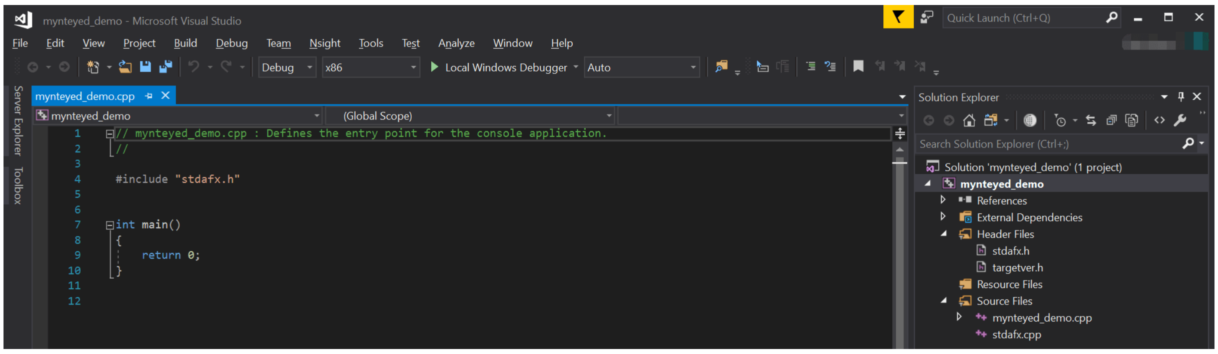
Location: C:\Users\John\AppData\Roaming\Slightech\MYNTEYED\SDK\1.3.6\projects\vs2017\

Solution name: mynteyed_demo

☑ Create directory for solution

☐ Create new Git repository

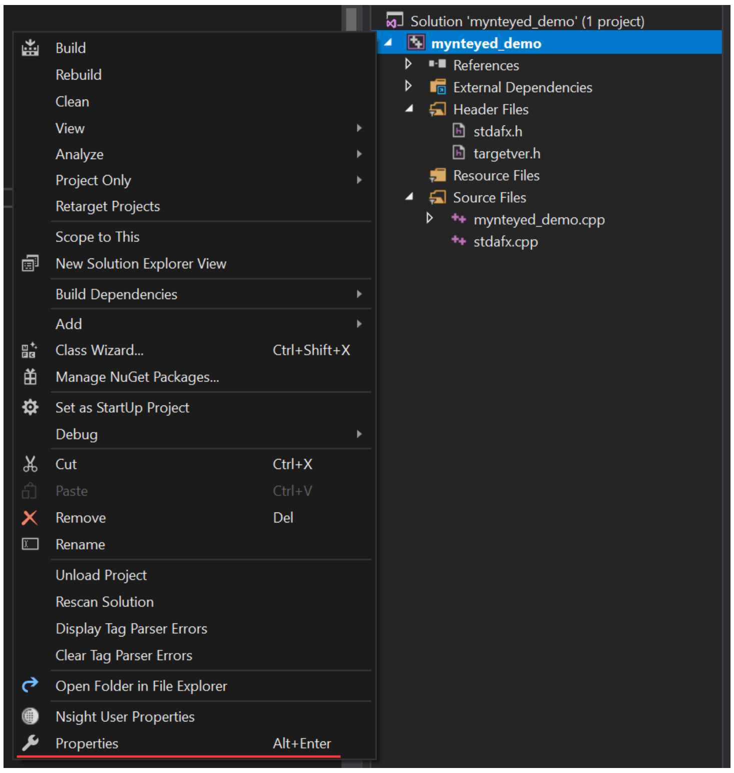
Finally, you will see the new project like this,



```
// mynteyed_demo.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"


int main()
{
    return 0;
}
```

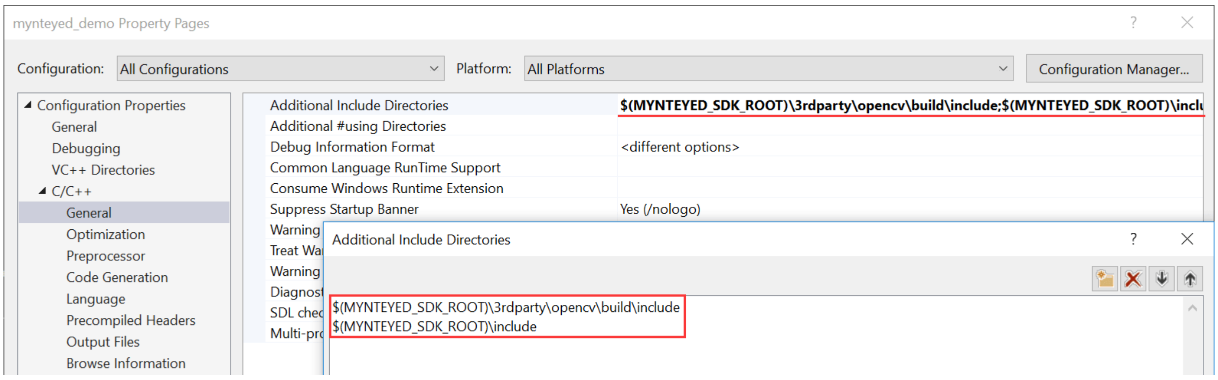
Config Properties

Right click the project, and open its "Properties" window,



Change "Configuration" to "All Configurations", then add the following paths to "Additional Include Directories",

```
$(MYNTEYED_SDK_ROOT)\include
$(MYNTEYED_SDK_ROOT)\3rdparty\opencv\build\include
```



Configuration: All Configurations Platform: All Platforms

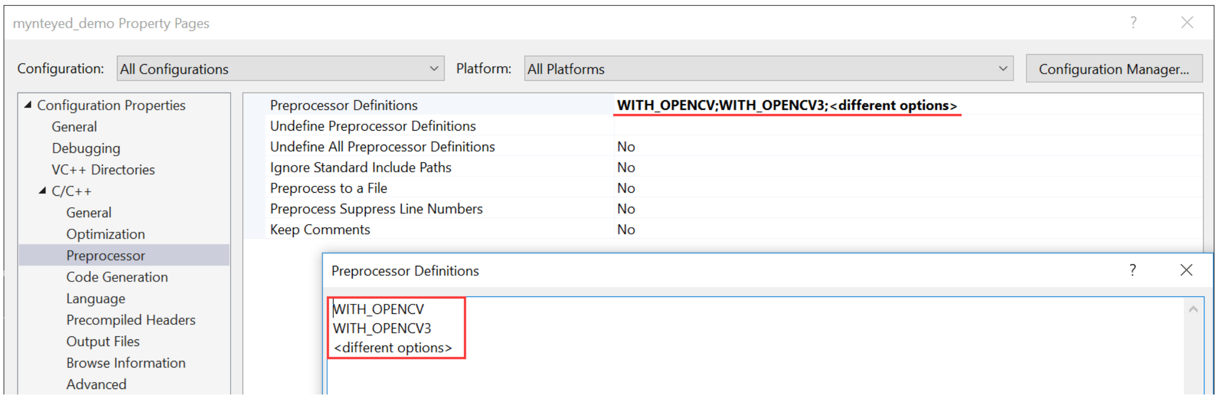
Additional Include Directories	$(MYNTEYED_SDK_ROOT)\3rdparty\opencv\build\include;$(MYNTEYED_SDK_ROOT)\inclu
Additional #using Directories	
Debug Information Format	<different options>
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)

Additional Include Directories

```
$(MYNTEYED_SDK_ROOT)\3rdparty\opencv\build\include
$(MYNTEYED_SDK_ROOT)\include
```

Add the following definitions to "Preprocessor Definitions",

```
WITH_OPENCV
WITH_OPENCV3
```



Configuration: All Configurations Platform: All Platforms

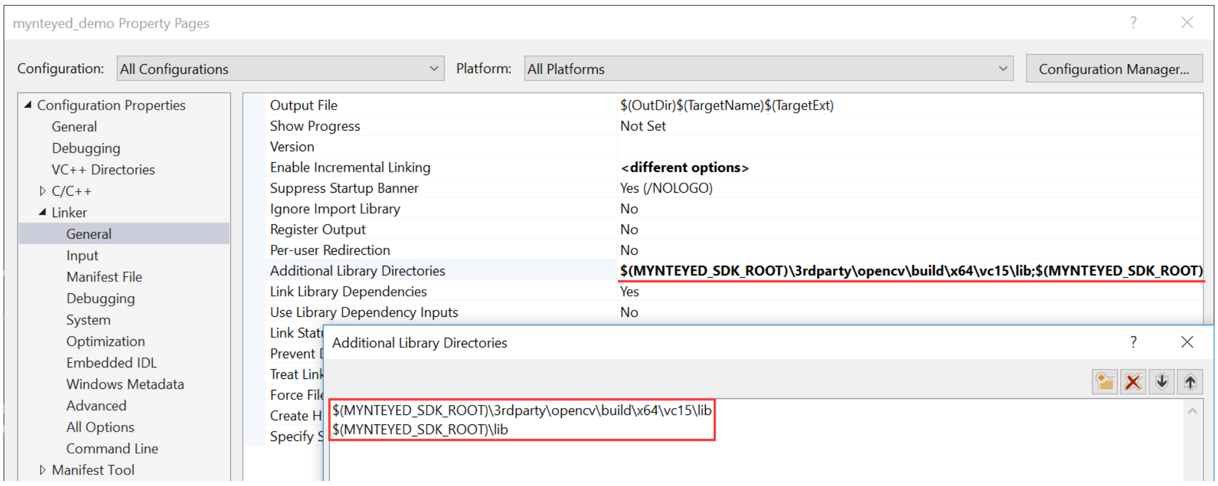
Preprocessor Definitions	WITH_OPENCV;WITH_OPENCV3;<different options>
Undefine Preprocessor Definitions	
Undefine All Preprocessor Definitions	No
Ignore Standard Include Paths	No
Preprocess to a File	No
Preprocess Suppress Line Numbers	No
Keep Comments	No

Preprocessor Definitions

```
WITH_OPENCV
WITH_OPENCV3
<different options>
```

Add the following paths to "Additional Library Directories",

```
$(MYNTEYED_SDK_ROOT)\lib
$(MYNTEYED_SDK_ROOT)\3rdparty\opencv\build\x64\vc15\lib
```



Configuration: All Configurations Platform: All Platforms

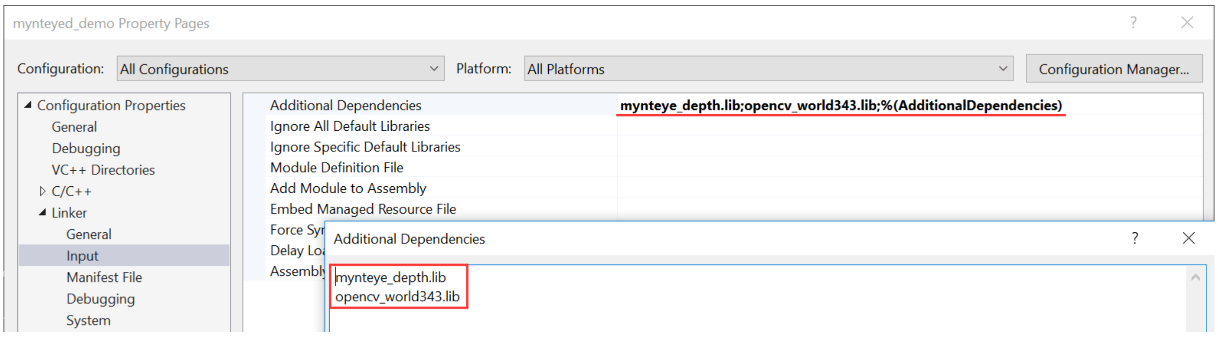
Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	**<different options>**
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	**$(MYNTEYED_SDK_ROOT)\3rdparty\opencv\build\x64\vc15\lib;$(MYNTEYED_SDK_ROOT)**
Link Library Dependencies	Yes
Use Library Dependency Inputs	No

Additional Library Directories

```
$(MYNTEYED_SDK_ROOT)\3rdparty\opencv\build\x64\vc15\lib
$(MYNTEYED_SDK_ROOT)\lib
```

Add the following libs to "Additional Dependencies",

```
mynteye_depth.lib
opencv_world343.lib
```



Configuration: All Configurations Platform: All Platforms

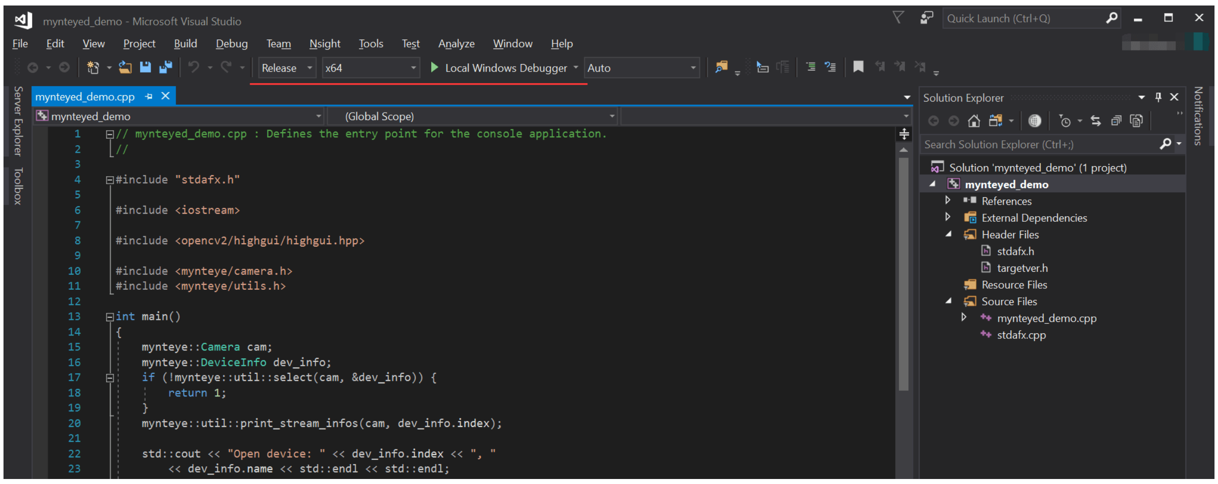
Additional Dependencies	**mynteye_depth.lib;opencv_world343.lib;%(AdditionalDependencies)**
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	

Additional Dependencies

```
mynteye_depth.lib
opencv_world343.lib
```

Start using SDK

Include the headers of SDK and start using its APIs,



```
// mynteyed_demo.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

#include <iostream>

#include <opencv2/highgui/highgui.hpp>

#include <mynteye/camera.h>
#include <mynteye/utils.h>

int main()
{
    mynteye::Camera cam;
    mynteye::DeviceInfo dev_info;
    if (!mynteye::util::select(cam, &dev_info)) {
        return 1;
    }
    mynteye::util::print_stream_infos(cam, dev_info.index);

    std::cout << "Open device: " << dev_info.index << ", "
        << dev_info.name << std::endl << std::endl;
```

Select "Release x64" to run the project.

6.2 How to use SDK with Qt Creator

This tutorial will create a Qt project with Qt Creator to start using SDK.

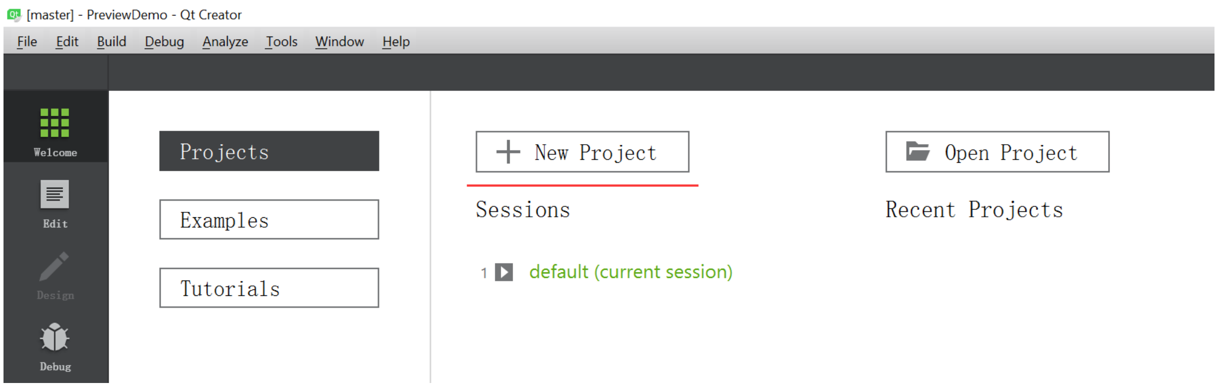
You could find the project demo in `<sdk>/platforms/projects/qtcreator` directory.

Preparation

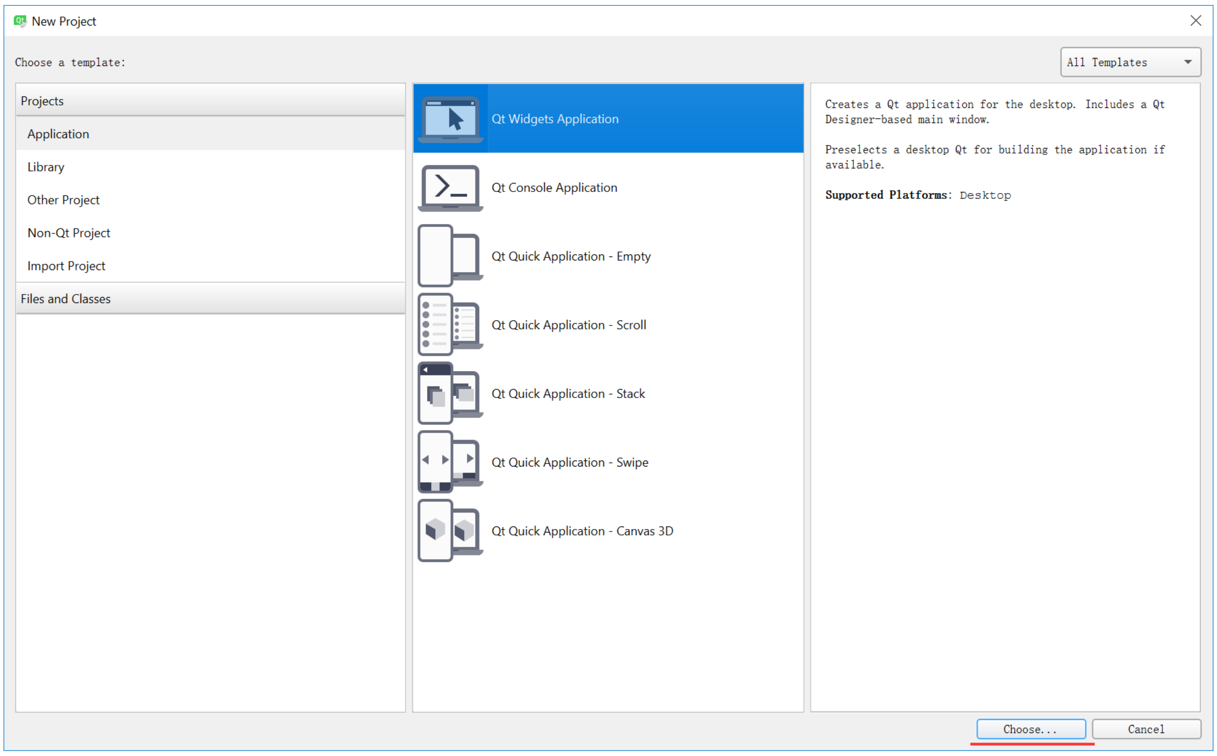
- Windows: install the win pack of SDK
- Linux: build from source and `make install`

Create Project

Open Qt Creator, then `New Project`,


[master] - PreviewDemo - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Welcome
Edit
Design
Debug
Projects
Examples
Tutorials
New Project
Open Project
Sessions
Recent Projects
1 default (current session)

Choose `Qt Widgets Application`,

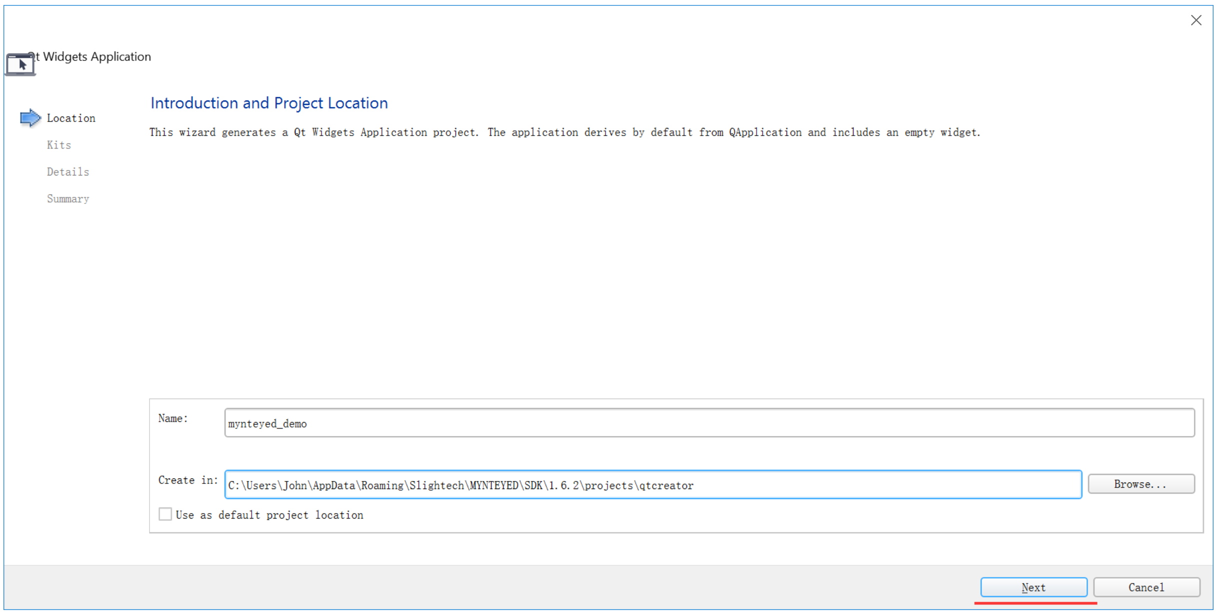


Creates a Qt application for the desktop. Includes a Qt Designer-based main window.

Preselects a desktop Qt for building the application if available.

Supported Platforms: Desktop

Set project location and its name,



Introduction and Project Location

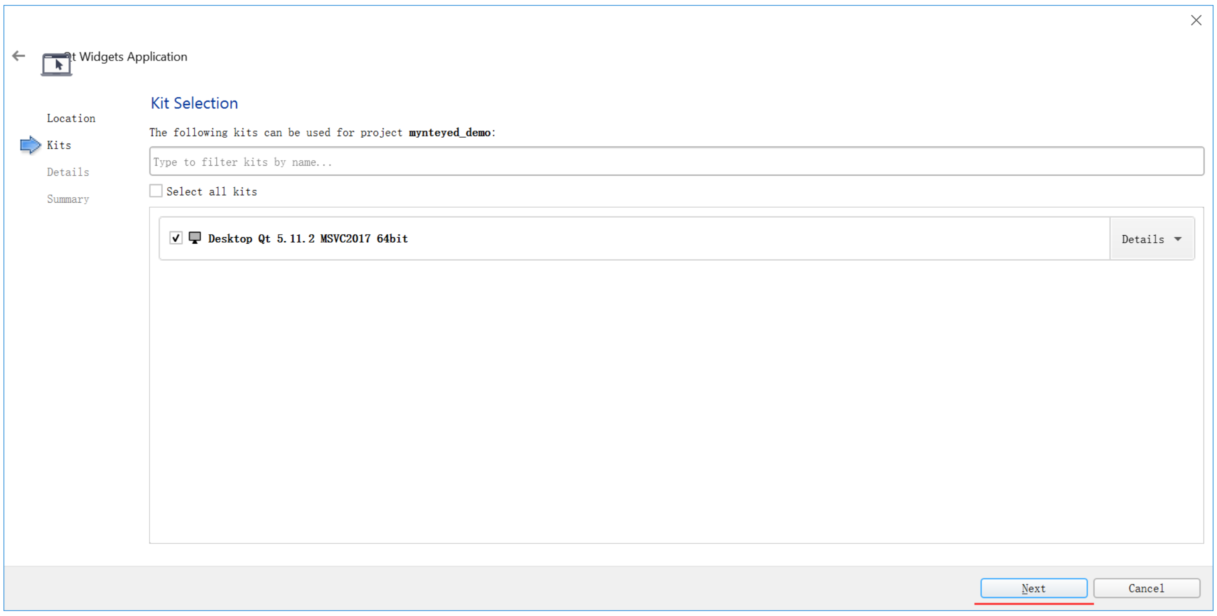
This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.

Name: mynteyed_demo

Create in: C:\Users\John\AppData\Roaming\Slightech\MYNTEYED\SDK\1.6.2\projects\qtcreator

☐ Use as default project location

Select the build kits,



Kit Selection

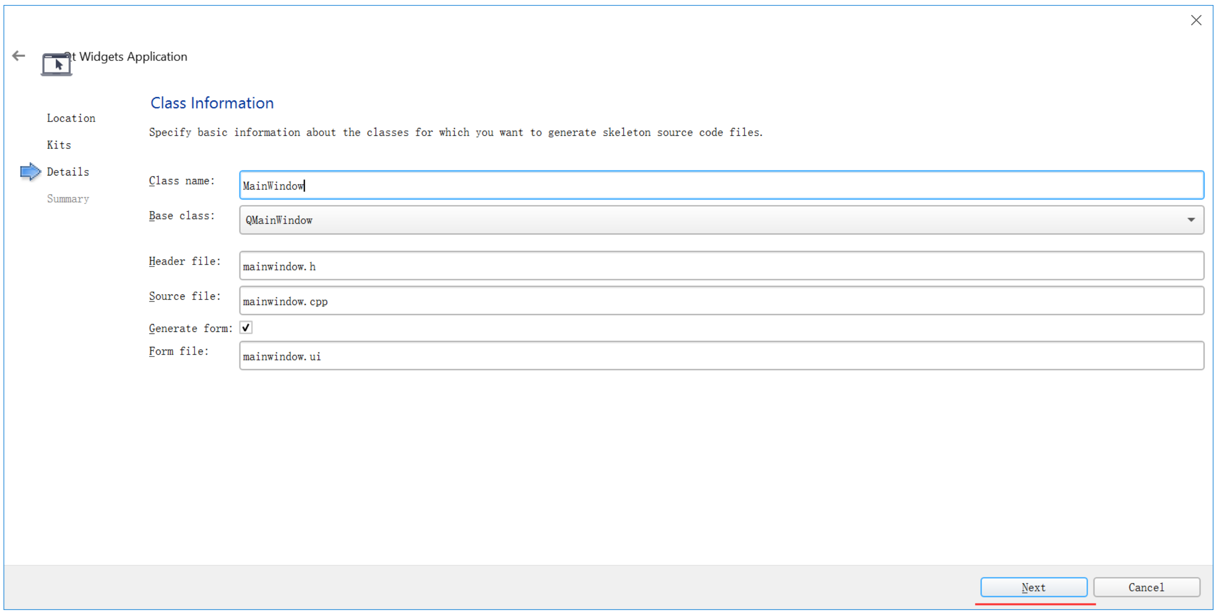
The following kits can be used for project **mynteyed_demo**:

Type to filter kits by name...

☐ Select all kits

☑ **Desktop Qt 5.11.2 MSVC2017 64bit**

Then, it will generate the skeleton source files,



Class Information

Specify basic information about the classes for which you want to generate skeleton source code files.

Class name: MainWindow

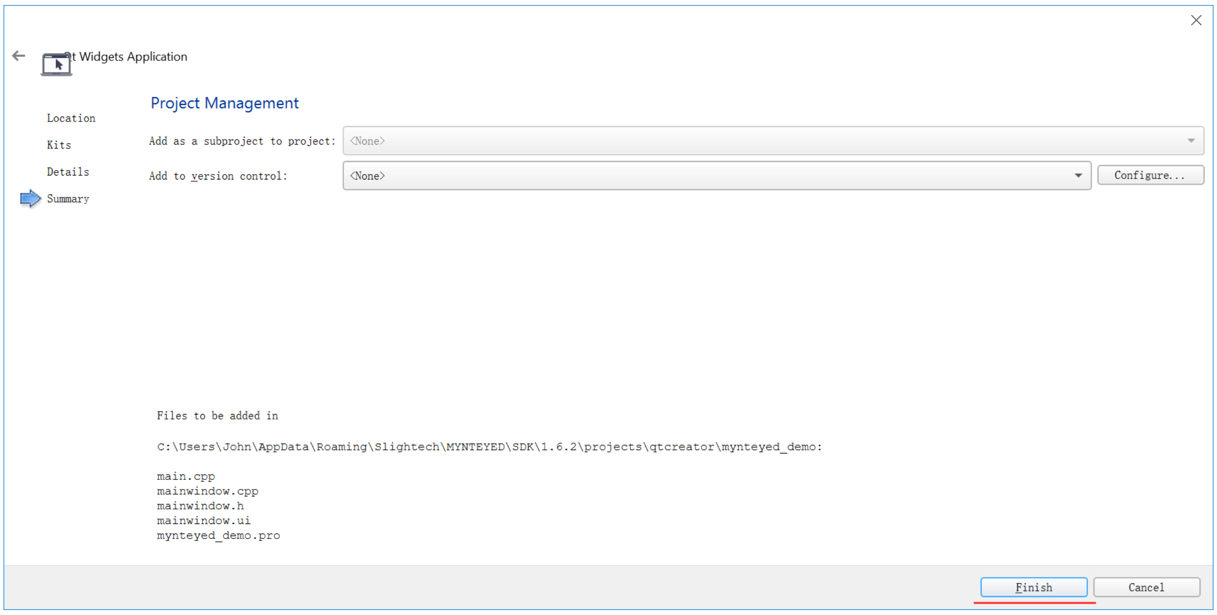
Base class: QMainWindow

Header file: mainwindow.h

Source file: mainwindow.cpp

Generate form: ✓

Form file: mainwindow.ui



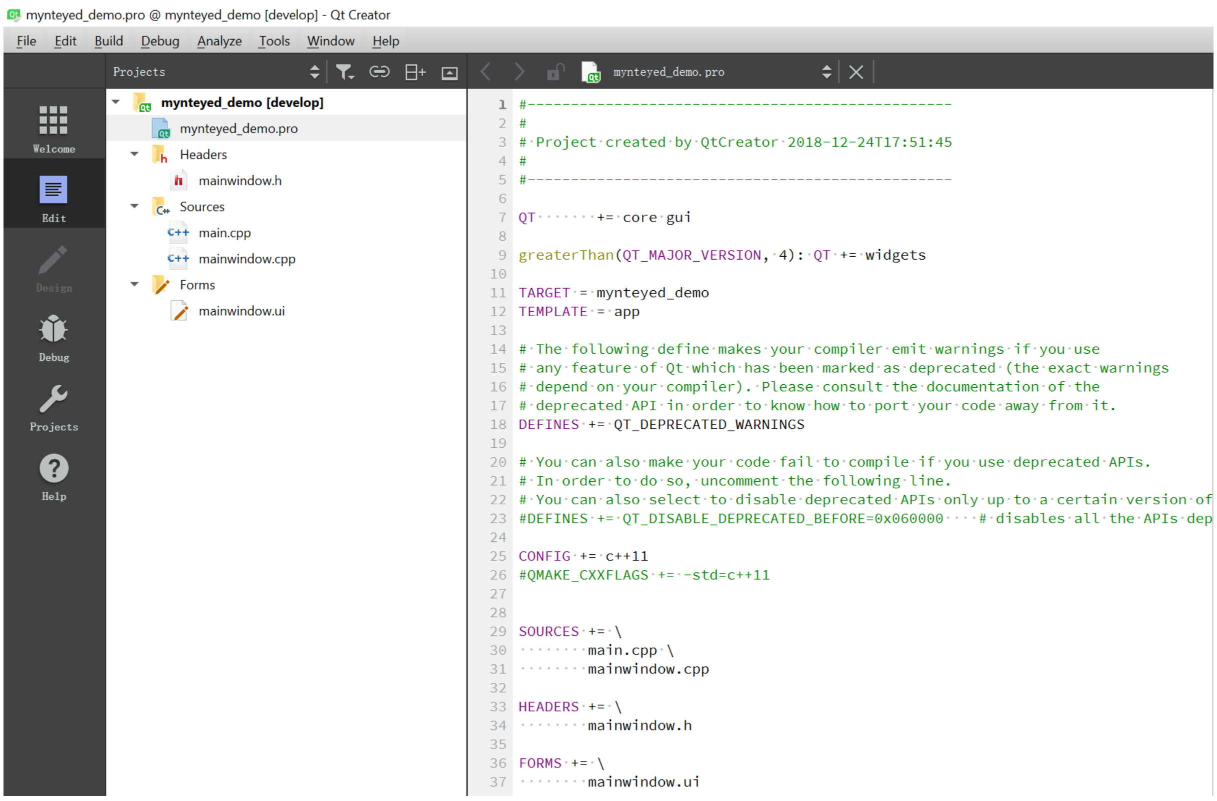
Project Management

Add as a subproject to project: <None>

Add to version control: <None>

```
Files to be added in

C:\Users\John\AppData\Roaming\Slightech\MYNTEYED\SDK\1.6.2\projects\qtcreator\mynteyed_demo:

main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
mynteyed_demo.pro
```

Finally, you will see the new project like this,



```
#-------------------------------------------------
#
# Project created by QtCreator 2018-12-24T17:51:45
#
#-------------------------------------------------

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = mynteyed_demo
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs dep

CONFIG += c++11
#QMAKE_CXXFLAGS += -std=c++11


SOURCES += \
        main.cpp \
        mainwindow.cpp

HEADERS += \
        mainwindow.h

FORMS += \
        mainwindow.ui
```

Config Project

Edit `mynteyed_demo.pro` to add `INCLUDEPATH` and `LIBS`.

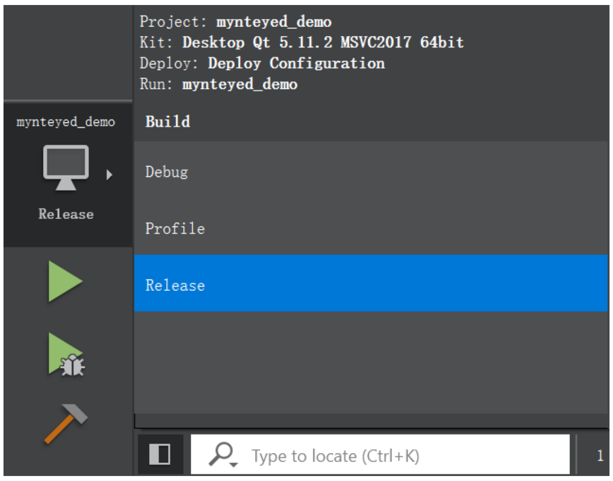
```
win32 {
    SDK_ROOT = "$$(MYNTEYED_SDK_ROOT)"
    isEmpty(SDK_ROOT) {
        error( "MYNTEYED_SDK_ROOT not found, please install SDK firstly" )
    }
    message("SDK_ROOT: $$SDK_ROOT")

    INCLUDEPATH += "$$SDK_ROOT/include"
    LIBS += "$$SDK_ROOT/lib/mynteye_depth.lib"
}

unix {
    INCLUDEPATH += /usr/local/include
    LIBS += -L/usr/local/lib -lmynteye_depth
}
```

Start using SDK

Include the headers of SDK and start using its APIs, could see the project demo.

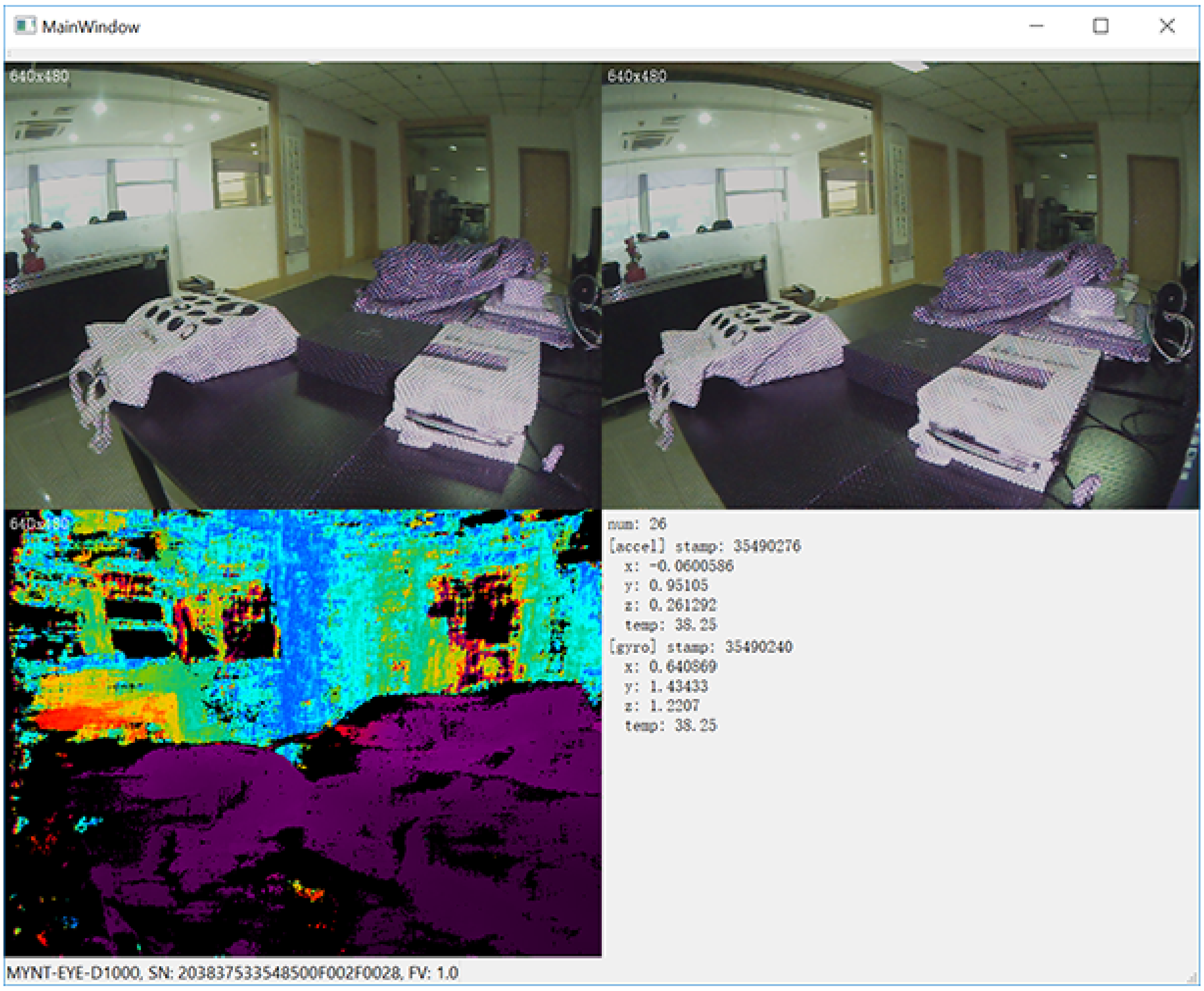
Windows

Should select "Release" to run the project.



```
Project: mynteyed_demo
Kit: Desktop Qt 5.11.2 MSVC2017 64bit
Deploy: Deploy Configuration
Run: mynteyed_demo
```

Then you will see the main window,

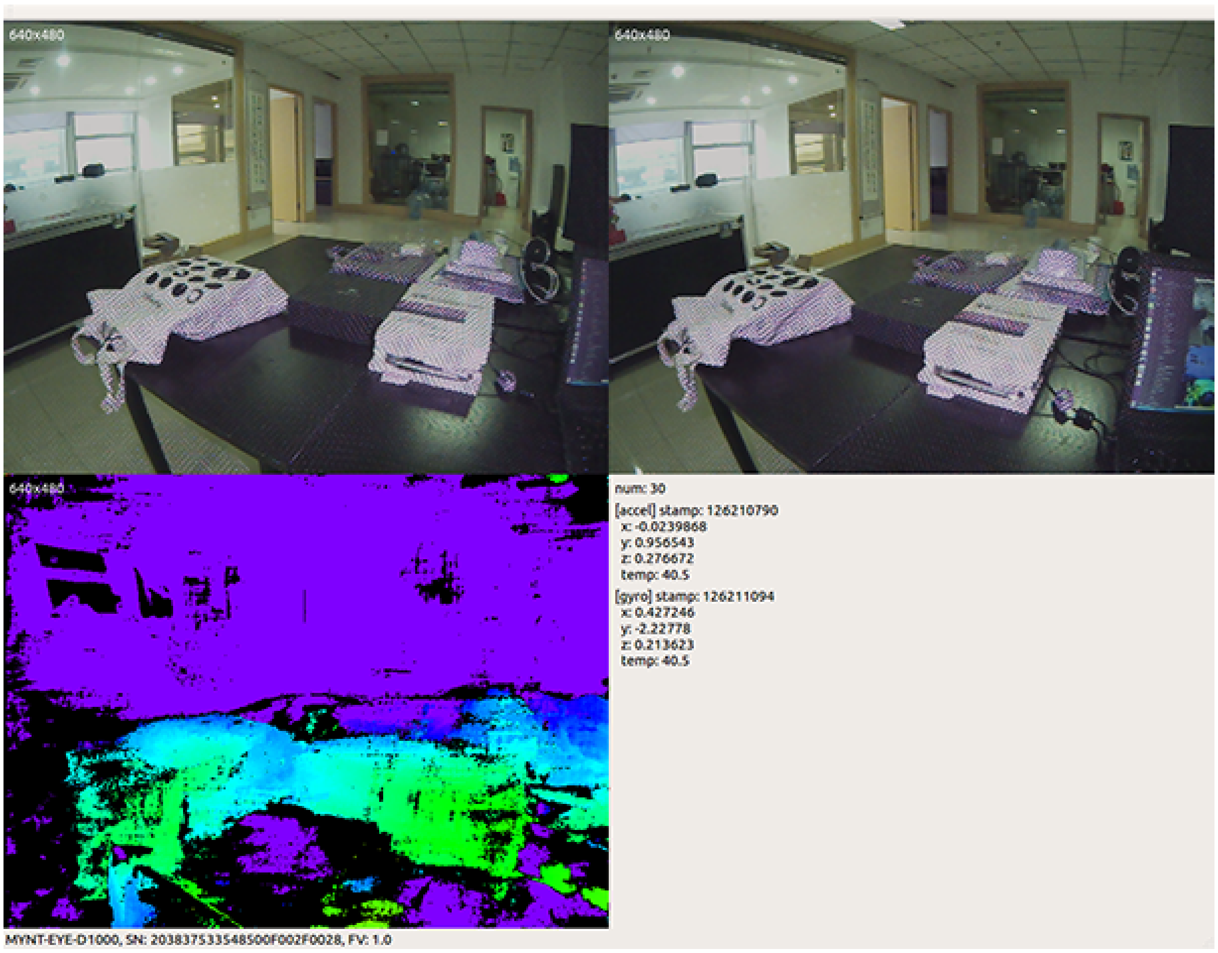


```
num: 26
[accel] stamp: 35490276
  x: -0.0600586
  y: 0.95105
  z: 0.261292
  temp: 38.25
[gyro] stamp: 35490240
  x: 0.640869
  y: 1.43433
  z: 1.2207
  temp: 38.25
```

MYNT-EYE-D1000, SN: 203837533548500F002F0028, FV: 1.0

Linux

Run the project and you will see the main window,



```
num: 30
[accel] stamp: 126210790
 x: -0.0239868
 y: 0.956543
 z: 0.276672
 temp: 40.5
[gyro] stamp: 126211094
 x: 0.427246
 y: -2.22778
 z: 0.213623
 temp: 40.5
```

```
MYNT-EYE-D1000, SN: 203837533548500F002F0028, FV: 1.0
```

6.3 How to use SDK with CMake

This tutorial will create a project with CMake to start using SDK.

> You could find the project demo in `<sdk>/platforms/projects/cmake` directory.

Preparation

- Windows: install the win pack of SDK
- Linux: build from source and `make install`

Create Project

Add `CMakeLists.txt` and `mynteyed_demo.cc` files,

```
cmake_minimum_required(VERSION 3.0)

project(mynteyed_demo VERSION 1.0.0 LANGUAGES C CXX)

# flags

set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wall -O3")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -O3")

set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -std=c++11 -march=native")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11 -march=native")

## mynteyed_demo

add_executable(mynteyed_demo mynteyed_demo.cc)
```

Config Project

Add `mynteyed` and `OpenCV` packages to `CMakeLists.txt`,

```
# packages

if(MSVC)
  set(SDK_ROOT "$ENV{MYNTEYED_SDK_ROOT}")
  if(SDK_ROOT)
    message(STATUS "MYNTEYED_SDK_ROOT: ${SDK_ROOT}")
    list(APPEND CMAKE_PREFIX_PATH
      "${SDK_ROOT}/lib/cmake"
      "${SDK_ROOT}/3rdparty/opencv/build"
    )
  else()
    message(FATAL_ERROR "MYNTEYED_SDK_ROOT not found, please install SDK firstly")
  endif()
endif()

## mynteyed

find_package(mynteyed REQUIRED)
message(STATUS "Found mynteye: ${mynteyed_VERSION}")

# When SDK build with OpenCV, we can add WITH_OPENCV macro to enable some
# features depending on OpenCV, such as ToMat().
if(mynteyed_WITH_OPENCV)
  add_definitions(-DWITH_OPENCV)
endif()

## OpenCV

# Set where to find OpenCV
#set(OpenCV_DIR "/usr/share/OpenCV")

# When SDK build with OpenCV, we must find the same version here.
find_package(OpenCV REQUIRED)
message(STATUS "Found OpenCV: ${OpenCV_VERSION}")
```

Add `include_directories` and `target_link_libraries` to `mynteyed_demo` target,

```
# targets

include_directories(
  ${OpenCV_INCLUDE_DIRS}
)

## mynteyed_demo

add_executable(mynteyed_demo mynteyed_demo.cc)
target_link_libraries(mynteyed_demo mynteye_depth ${OpenCV_LIBS})
```

Start using SDK

Include the headers of SDK and start using its APIs, could see the project demo.

Windows

See Quick Start Guide for Windows to "Install Build Tools".

Then open "x64 Native Tools Command Prompt for VS 2017" command shell to build and run,

```
mkdir _build
cd _build

cmake -G "Visual Studio 15 2017 Win64" ..

msbuild.exe ALL_BUILD.vcxproj /property:Configuration=Release

.\Release\mynteyed_demo.exe
```

Linux

Open "Terminal" to build and run,

```
mkdir _build
cd _build/

cmake ..

make

./mynteyed_demo
```

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 8

Class Index

8.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 9

Class Documentation

9.1 mynteyed::Camera Class Reference

Public Member Functions

- std::vector< DeviceInfo > GetDeviceInfos () const

  Get all device infos.
- void GetDeviceInfos (std::vector< DeviceInfo > ∗dev_infos) const

  Get all device infos.
- void GetStreamInfos (const std::int32_t &dev_index, std::vector< StreamInfo > ∗color_infos, std::vector< StreamInfo > ∗depth_infos) const

  Get all stream infos.
- ErrorCode Open ()

  Open camera.
- ErrorCode Open (const OpenParams &params)

  Open camera with params.
- bool IsOpened () const

  Whethor camera is opened or not.
- std::shared_ptr< device::Descriptors > GetDescriptors () const

  Get all device descriptors.
- std::string GetDescriptor (const Descriptor &desc) const

  Get one device descriptor.
- StreamIntrinsics GetStreamIntrinsics (const StreamMode &stream_mode) const

  Get the intrinsics of camera.
- StreamIntrinsics GetStreamIntrinsics (const StreamMode &stream_mode, bool ∗ok) const

  Get the intrinsics of camera.
- StreamExtrinsics GetStreamExtrinsics (const StreamMode &stream_mode) const

  Get the extrinsics of camera.
- StreamExtrinsics GetStreamExtrinsics (const StreamMode &stream_mode, bool ∗ok) const

  Get the extrinsics of camera.
- bool WriteCameraCalibrationBinFile (const std::string &filename)

  Write camera calibration bin file.
- MotionIntrinsics GetMotionIntrinsics () const

  Get the intrinsics of motion.
- MotionIntrinsics GetMotionIntrinsics (bool ∗ok) const

  Get the intrinsics of motion.

- MotionExtrinsics GetMotionExtrinsics () const

  Get the extrinsics from left to motion.
- MotionExtrinsics GetMotionExtrinsics (bool ∗ok) const

  Get the extrinsics from left to motion.
- bool IsWriteDeviceSupported () const

  Whethor write device supported or not.
- bool WriteDeviceFlash (device::Descriptors ∗desc, device::ImuParams ∗imu_params, Version ∗spec_↩ version=nullptr)

  Write device flash.
- void EnableProcessMode (const ProcessMode &mode)

  Enable process mode, e.g.
- void EnableProcessMode (const std::int32_t &mode)

  Enable process mode, e.g.
- bool IsImageInfoSupported () const

  Whethor image info supported or not.
- void EnableImageInfo (bool sync)

  Enable image infos.
- void DisableImageInfo ()

  Disable image info.
- bool IsImageInfoEnabled () const

  Whethor image info enabled or not.
- bool IsImageInfoSynced () const

  Whethor image info synced or not.
- bool IsStreamDataEnabled (const ImageType &type) const

  Whethor stream data of certain image type enabled or not.
- bool HasStreamDataEnabled () const

  Has any stream data enabled.
- StreamData GetStreamData (const ImageType &type)

  Get latest stream data.
- std::vector< StreamData > GetStreamDatas (const ImageType &type)

  Get cached stream datas.
- bool IsMotionDatasSupported () const

  Whethor motion datas supported or not.
- void EnableMotionDatas (std::size_t max_size=std::numeric_limits< std::size_t >::max())

  Enable motion datas.
- void DisableMotionDatas ()

  Disable motion datas.
- bool IsMotionDatasEnabled () const

  Whethor motion datas enabled or not.
- std::vector< MotionData > GetMotionDatas ()

  Get cached motion datas.
- void SetImgInfoCallback (img_info_callback_t callback, bool async=true)

  Set image info callback.
- void SetStreamCallback (const ImageType &type, stream_callback_t callback, bool async=true)

  Set stream data callback.
- void SetMotionCallback (motion_callback_t callback, bool async=true)

  Set motion data callback.
- void Close ()

  Close the camera.
- bool HidFirmwareUpdate (const char ∗filepath)

  Update hid device firmware.

- void SetExposureTime (const float &value)

  Set exposure time [1ms - 2000ms] value – exposure time value.
- void GetExposureTime (float &value)

  Get exposure time value – return exposure time value.
- void SetGlobalGain (const float &value)

  Set global gain [1 - 16] value – global gain value.
- void GetGlobalGain (float &value)

  Get global gain value – return global gain value.
- void SetIRIntensity (const std::uint16_t &value)

  set infrared(IR) intensity [0, 10] default 4
- bool AutoExposureControl (bool enable)

  Auto-exposure enabled or not default enabled.
- bool AutoWhiteBalanceControl (bool enable)

  Auto-white-balance enabled or not default enabled.

9.1.1 Member Function Documentation

9.1.1.1 void mynteyed::Camera::DisableImageInfo ( )

Disable image info.

9.1.1.2 void mynteyed::Camera::DisableMotionDatas ( )

Disable motion datas.

9.1.1.3 void mynteyed::Camera::EnableImageInfo (bool *sync*)

Enable image infos.

If sync is false, indicates only can get infos from callback. If sync is true, indicates can get infos from callback or access it from StreamData.

9.1.1.4 void mynteyed::Camera::EnableMotionDatas (std::size_t *max_size* = `std::numeric_limits< std:↩ :size_t >::max()`)

Enable motion datas.

If max_size <= 0, indicates only can get datas from callback. If max_size > 0, indicates can get datas from callback or using GetMotionDatas().

Note: if max_size > 0, the motion datas will be cached until you call GetMotionDatas().

9.1.1.5 void mynteyed::Camera::EnableProcessMode (const ProcessMode & *mode*)

Enable process mode, e.g.

imu assembly, temp_drift

9.1.1.6 void mynteyed::Camera::EnableProcessMode (const std::int32_t & *mode*)

Enable process mode, e.g.

imu assembly, temp_drift

9.1.1.7 std::vector< MotionData > mynteyed::Camera::GetMotionDatas ( )

Get cached motion datas.

Besides, you can also get them from callback

9.1.1.8 void mynteyed::Camera::SetImgInfoCallback (img_info_callback_t *callback*, bool *async* = true)

Set image info callback.

9.1.1.9 void mynteyed::Camera::SetMotionCallback (motion_callback_t *callback*, bool *async* = true)

Set motion data callback.

9.1.1.10 void mynteyed::Camera::SetStreamCallback (const ImageType & *type*, stream_callback_t *callback*, bool *async* = true)

Set stream data callback.

9.2 mynteyed::CameraIntrinsics Struct Reference

Camera intrinsics: size, coeffs and camera matrix.

Public Attributes

- std::uint16_t width

  The width of the image in pixels.
- std::uint16_t height

  The height of the image in pixels.
- double fx

  The focal length of the image plane, as a multiple of pixel width.
- double fy

  The focal length of the image plane, as a multiple of pixel height.
- double cx

  The horizontal coordinate of the principal point of the image.
- double cy

  The vertical coordinate of the principal point of the image.
- double coeffs [5]

  The distortion coefficients: k1,k2,p1,p2,k3.
- double p [12]

  3x4 projection matrix in the (rectified) coordinate systems left: fx' cx' fy' cy' 1 right: fx' cx' tx fy' cy' 1

9.2.1 Detailed Description

Camera intrinsics: size, coeffs and camera matrix.

9.3 mynteyed::device::Descriptors Struct Reference

Device descriptors.

9.3.1 Detailed Description

Device descriptors.

9.4 mynteyed::DeviceInfo Struct Reference

Device information.

Public Attributes

- std::int32_t index

  The device index.
- std::string name

  The device name.
- std::uint16_t type

  The device type.
- std::uint16_t pid

  The product id.
- std::uint16_t vid

  The vendor id.
- std::uint16_t chip_id

  The chip id.
- std::string fw_version

  The firmware version.

9.4.1 Detailed Description

Device information.

9.5 mynteyed::Extrinsics Struct Reference

Extrinsics, represent how the different datas are connected.

Public Member Functions

- Extrinsics Inverse () const

  Inverse this extrinsics.

Public Attributes

- double rotation [3][3]

  Rotation matrix left camera to right camera.
- double translation [3]

  Translation vector left camera to right camera.

9.5.1 Detailed Description

Extrinsics, represent how the different datas are connected.

9.5.2 Member Function Documentation

9.5.2.1 Extrinsics mynteyed::Extrinsics::Inverse ( ) const [inline]

Inverse this extrinsics.

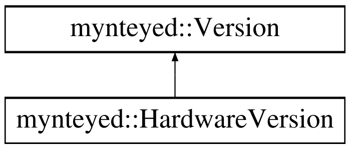
Returns

the inversed extrinsics.

9.6 mynteyed::HardwareVersion Class Reference

Hardware version.

Inheritance diagram for mynteyed::HardwareVersion:

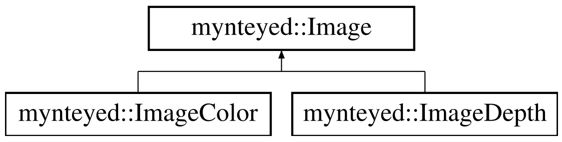


9.6.1 Detailed Description

Hardware version.

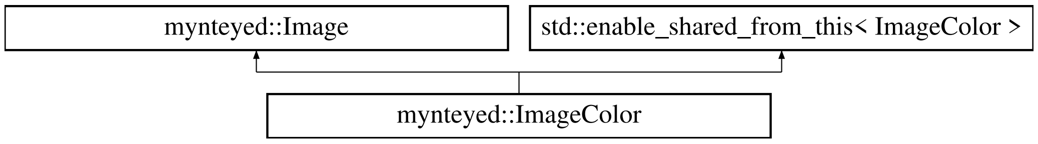
9.7 mynteyed::Image Class Reference

Inheritance diagram for mynteyed::Image:



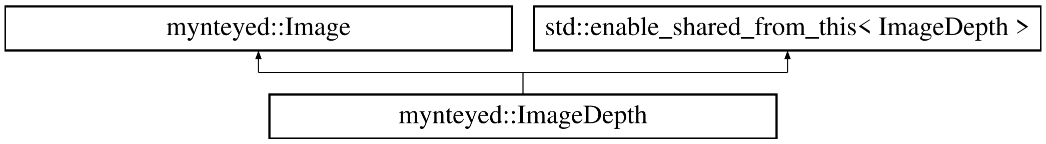
9.8 mynteyed::ImageColor Class Reference

Inheritance diagram for mynteyed::ImageColor:



9.9 mynteyed::ImageDepth Class Reference

Inheritance diagram for mynteyed::ImageDepth:



9.10 mynteyed::ImgInfo Struct Reference

Image information.

Public Attributes

- std::uint16_t frame_id

  Image frame id.
- std::uint32_t timestamp

  Image timestamp.
- std::uint16_t exposure_time

  Image exposure time.

9.10.1 Detailed Description

Image information.

9.11 mynteyed::ImuData Struct Reference

Imu data.

Public Attributes

- std::uint8_t flag

  Data type MYNTEYE_IMU_ACCEL: accelerometer MYNTEYE_IMU_GYRO: gyroscope.
- std::uint64_t timestamp

  Imu gyroscope or accelerometer or frame timestamp.
- double temperature

  temperature
- double accel [3]

  Imu accelerometer data for 3-axis: X, Y, X.
- double gyro [3]

  Imu gyroscope data for 3-axis: X, Y, Z.

9.11.1 Detailed Description

Imu data.

9.11.2 Member Data Documentation

9.11.2.1 double mynteyed::ImuData::accel[3]

Imu accelerometer data for 3-axis: X, Y, X.

9.11.2.2 double mynteyed::ImuData::gyro[3]

Imu gyroscope data for 3-axis: X, Y, Z.

9.12 mynteyed::ImuIntrinsics Struct Reference

IMU intrinsics: scale, drift and variances.

Public Attributes

- double scale [3][3]

  Scale matrix.
- double assembly [3][3]

  Assembly error [3][3].
- double noise [3]

  Noise density variances.
- double bias [3]

  Random walk variances.
- double x [2]

  Temperature drift.

9.12.1 Detailed Description

IMU intrinsics: scale, drift and variances.

9.12.2 Member Data Documentation

9.12.2.1 double mynteyed::ImuIntrinsics::scale[3][3]

Scale matrix.

```
Scale X     cross axis  cross axis
cross axis  Scale Y     cross axis
cross axis  cross axis  Scale Z
```

9.12.2.2 double mynteyed::ImuIntrinsics::x[2]

Temperature drift.

```
0 - Constant value
1 - Slope
```

9.13 mynteyed::device::ImuParams Struct Reference

Device imu paramters.

9.13.1 Detailed Description

Device imu paramters.

9.14 mynteyed::MotionData Struct Reference

Motion data.

Public Attributes

- std::shared_ptr< ImuData > imu

    ImuData.

9.14.1 Detailed Description

Motion data.

9.14.2 Member Data Documentation

9.14.2.1 std::shared_ptr<ImuData> mynteyed::MotionData::imu

ImuData.

9.15 mynteyed::MotionIntrinsics Struct Reference

Motion intrinsics, including accelerometer and gyroscope.

Public Attributes

- ImuIntrinsics accel

    Accelerometer intrinsics.
- ImuIntrinsics gyro

    Gyroscope intrinsics.

9.15.1 Detailed Description

Motion intrinsics, including accelerometer and gyroscope.

9.16 mynteyed::OpenParams Struct Reference

Device open parameters.

Public Member Functions

- OpenParams ()

  Constructor.
- ∼OpenParams ()

  Destructor.

Public Attributes

- std::int32_t dev_index

  Device index.
- std::int32_t framerate

  Framerate, range [0,60], [0,30](STREAM_2560x720), default 10.
- DeviceMode dev_mode

  Device mode, default DEVICE_ALL.
- ColorMode color_mode

  Color mode, default COLOR_RAW.
- DepthMode depth_mode

  Depth mode, default DEPTH_COLORFUL.
- StreamMode stream_mode

  Stream mode of color & depth, default STREAM_1280x720.
- StreamFormat color_stream_format

  Stream format of color, default STREAM_YUYV.
- StreamFormat depth_stream_format

  Stream format of depth, default STREAM_YUYV.
- bool state_ae

  Auto-exposure, default true.
- bool state_awb

  Auto-white balance, default true.
- std::uint8_t ir_intensity

  IR (Infrared), range [0,10], default 0.
- bool ir_depth_only

  IR Depth Only mode, default false.
- float colour_depth_value

  Colour depth image, default 5000.

9.16.1 Detailed Description

Device open parameters.

9.16.2 Constructor & Destructor Documentation

9.16.2.1 mynteyed::OpenParams::OpenParams ( )

Constructor.

9.16.2.2 mynteyed::OpenParams::∼OpenParams ( )

Destructor.

9.16.3 Member Data Documentation

9.16.3.1 float mynteyed::OpenParams::colour_depth_value

Colour depth image, default 5000.

[0, 16384]

9.16.3.2 DeviceMode mynteyed::OpenParams::dev_mode

Device mode, default DEVICE_ALL.

- DEVICE_COLOR: IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH n
- DEVICE_DEPTH: IMAGE_LEFT_COLOR n IMAGE_RIGHT_COLOR n IMAGE_DEPTH y
- DEVICE_ALL: IMAGE_LEFT_COLOR y IMAGE_RIGHT_COLOR - IMAGE_DEPTH y

Could detect image type is enabled after opened through Camera::IsStreamDataEnabled().

Note: y: available, n: unavailable, -: depends on stream_mode

9.16.3.3 bool mynteyed::OpenParams::ir_depth_only

IR Depth Only mode, default false.

Note: When frame rate less than 30fps, IR Depth Only will be not available.

9.17 mynteyed::Rate Class Reference

9.18 mynteyed::StreamData Struct Reference

Stream data.

Public Attributes

- std::shared_ptr< Image > img
  Image data.
- std::shared_ptr< ImgInfo > img_info
  Image information.

9.18.1 Detailed Description

Stream data.

9.19 mynteyed::StreamInfo Struct Reference

Stream information.

Public Attributes

- std::int32_t index

  The stream index.
- std::int32_t width

  The stream width.
- std::int32_t height

  The stream height.
- StreamFormat format

  The stream format.

9.19.1 Detailed Description

Stream information.

9.20 mynteyed::StreamIntrinsics Struct Reference

Camera intrinsics: size, coeffs and camera matrix.

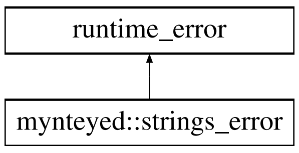
9.20.1 Detailed Description

Camera intrinsics: size, coeffs and camera matrix.

9.21 mynteyed::strings_error Class Reference

The strings error.

Inheritance diagram for mynteyed::strings_error:



9.21.1 Detailed Description

The strings error.

9.22 mynteyed::Type Class Reference

Type.

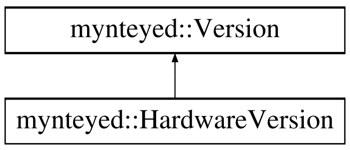
9.22.1 Detailed Description

Type.

9.23 mynteyed::Version Class Reference

Version.

Inheritance diagram for mynteyed::Version:



9.23.1 Detailed Description

Version.

Index